```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics;
namespace First_Application
{
    public partial class FormSimulation : Form
    {
        public FormSimulation()
        {
            InitializeComponent();
        }
        public bool power;
        public bool geography;
        public Random rnd = new Random();
        public int x;
        public int y;
        public int number = 0;
        public double summa = 0;
        public double temptation;
        public double reward;
        public double punishment;
        public double sucker;
        public double maxDistance;
        public double misconduct;
        public double misperception;
        public double payoffshift;
        public DataTable Strategies = new DataTable();
        public DataTable Inputs = new DataTable();
        public DataTable Output = new DataTable();
        public DataTable Cooperation = new DataTable();
        public DataTable Structures = new DataTable();
        public int[,] coop;         // number of mutually cooperative outcomes of
    interactions of any two players
        public int[] struc;    // number of interactions of any two actors; possible emergent
     structure
        public double[,] gain;
        public double[,] input;
        public double[] outputs;
        public bool strategyA;
        public bool strategyB;
        List<double[]> outcomes = new List<double[]>();
        List<int[]> structure = new List<int[]>();
        class Actors
        {
            public double score;
            public int strategy;
            public int row;
            public int column;
            public double[] standing;    // 1 = content [c], 2 = provoked [p], 3 = contrite
    [r]
            public double[] rewardshift; // approaching temptation after mutual cooperation,
    and punishment otherwise
            public double[] encounters;  // number of interactions between particular two
    players
            public double[] defections;  // number of other player's defections
            public double[][,] array;
            public double[] paramOne;
            public double[] paramTwo;
            public double[][] variables;

            public List<int[]> interactions; // list of interactions in all rounds
            public int[] interaction;   // given round interactions
            // "" or no interaction would be 99; or 0
```

```
            // "11" or CC would be 11; or 1
            // "10" or CD would be 10; or 2
            // "01" or DC would be 1;  or 3
            // "00" or DD would be 0;  or 4
        }
        Actors[] actors;
        private void FormSimulation_Load(object sender, EventArgs e)
        {
            int g = 0;
            btnWinner.Enabled = false;
            coop = new int[x * y, x * y];
            struc = new int[x * y * (x * y - 1) / 2];
            input = new double[x, y];
            outputs = new double[x * y];
            actors = new Actors[x * y];
            for (int n = 0; n < x * y; n++)
            {
                input[n / y, n % y] = Double.Parse(Inputs.Rows[n / y][n % y].ToString());
                actors[n] = new Actors();
                actors[n].strategy = Int32.Parse(Strategies.Rows[n / y][n % y].ToString());
                actors[n].score = input[n / y, n % y];
                actors[n].row = n / y;
                actors[n].column = n % y;
                actors[n].interactions = new List<int[]>();
                actors[n].interaction = new int[x * y];
                actors[n].encounters = new double[x * y];
                actors[n].defections = new double[x * y];
                actors[n].paramOne = new double[x * y];
                actors[n].paramTwo = new double[x * y];
                actors[n].rewardshift = new double[x * y];
                actors[n].standing = new double[x * y];
                actors[n].array = new double[x * y][,];
                actors[n].variables = new double[x * y][];
                for (int m = 0; m < x * y; m++)
                {
                    // other parameters are automatically initialized to the default value of ↙
 0
                    if (m > n)
                    {
                        struc[g] = n + 1;
                        g++;
                    }
                    actors[n].standing[m] = 1;      // standing in 1st round is content = 1   ↙

                    actors[n].array[m] = new double[2, 2];
                    actors[n].variables[m] = new double[100];
                }
                if (actors[n].strategy == 80)
                {
                    actors[n].standing[0] = Math.Round(rnd.NextDouble(), 1);    //            ↙
probability of cooperation after CC
                    actors[n].standing[1] = Math.Round(rnd.NextDouble(), 1);    //            ↙
probability of cooperation after CD
                    actors[n].standing[2] = Math.Round(rnd.NextDouble(), 1);    //            ↙
probability of cooperation after DC
                    actors[n].standing[3] = Math.Round(rnd.NextDouble(), 1);    //            ↙
probability of cooperation after DD
                }
                Output.Columns.Add();
                Cooperation.Columns.Add();
                if (actors[n].strategy == 80)
                {
                    if (actors[n].standing[0] == 0 && actors[n].standing[1] == 0 && actors[n]↙
.standing[2] == 0)
                    {
                        if (actors[n].standing[3] == 1)
                            outputs[n] = 9999910;
                        else
                            outputs[n] = 9999990 + actors[n].standing[3] * 10;
```

```
                }
                else
                    outputs[n] = actors[n].standing[0] * 10000000 + actors[n].standing[1] ↙
 * 100000 + actors[n].standing[2] * 1000 + actors[n].standing[3] * 10;
            }
            else
                outputs[n] = actors[n].strategy;
        }
        g = 0;
        structure.Add(struc);
        struc = new int[x * y * (x * y - 1) / 2];
        for (int n = 0; n < x * y; n++)
        {
            object[] row = new object[x * y];
            for (int m = 0; m < x * y; m++)
            {
                if (m > n)
                {
                    struc[g] = m + 1;
                    g++;
                }
                row[m] = 0;
                coop[n, m] = 0;
                actors[n].interaction[m] = actors[m].strategy;
            }
            Cooperation.Rows.Add(row);
            actors[n].interactions.Add(actors[n].interaction);
        }
        structure.Add(struc);
        struc = new int[x * y * (x * y - 1) / 2];
        maxDistance = Math.Sqrt(Math.Pow(x / 2, 2) + Math.Pow(y / 2, 2));
        Cooperation.AcceptChanges();
        Inputs.AcceptChanges();
        Output.AcceptChanges();
        Structures.AcceptChanges();
        dgvResults.DataSource = Inputs;
        outcomes.Add(outputs);
    }
    private void action(int n, int m)
    {
        if (strategyA == true && strategyB == true)     //Mutual cooperation CC
        {
            if (actors[n].rewardshift[m] < 0)
            {
                gain[n / y, n % y] += (reward - (reward - punishment) * (1 - Math.Pow ↙
(payoffshift, Math.Abs(actors[n].rewardshift[m]))));
                gain[m / y, m % y] += (reward - (reward - punishment) * (1 - Math.Pow ↙
(payoffshift, Math.Abs(actors[m].rewardshift[n]))));
            }
            else
            {
                gain[n / y, n % y] += (reward + (temptation - reward) * (1 - Math.Pow ↙
(payoffshift, actors[n].rewardshift[m])));
                gain[m / y, m % y] += (reward + (temptation - reward) * (1 - Math.Pow ↙
(payoffshift, actors[m].rewardshift[n])));
            }
            actors[n].rewardshift[m]++;
            actors[m].rewardshift[n]++;
            coop[n, m]++;    // counting CC for each pair
            coop[m, n]++;
            actors[n].interaction[m] = 1;
            actors[m].interaction[n] = 1;
        }
        else if (strategyA == true && strategyB == false)   //Unilateral cooperation CD
        {
            actors[n].rewardshift[m]--;
            actors[m].rewardshift[n]--;
            gain[n / y, n % y] += sucker;
            gain[m / y, m % y] += temptation;
```

```
                actors[n].interaction[m] = 2;
                actors[m].interaction[n] = 3;
            }
            else if (strategyA == false && strategyB == true)   //Unilateral defection DC
            {
                actors[n].rewardshift[m]--;
                actors[m].rewardshift[n]--;
                gain[n / y, n % y] += temptation;
                gain[m / y, m % y] += sucker;
                actors[n].interaction[m] = 3;
                actors[m].interaction[n] = 2;
            }
            else        //Mutual Defection DD
            {
                actors[n].rewardshift[m]--;
                actors[m].rewardshift[n]--;
                gain[n / y, n % y] += punishment;
                gain[m / y, m % y] += punishment;
                actors[n].interaction[m] = 4;
                actors[m].interaction[n] = 4;
            }
        }
        private void mispercieve(int No, int Adv)  //Misperception introduction
        {
            for (int a = actors[No].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[No].interactions[a][Adv] != 0)
                {
                    if (actors[No].interactions[a][Adv] == 1)
                        actors[No].interactions[a][Adv] = 2;
                    else if (actors[No].interactions[a][Adv] == 3)
                        actors[No].interactions[a][Adv] = 4;
                    else if (actors[No].interactions[a][Adv] == 2)
                        actors[No].interactions[a][Adv] = 1;
                    else
                        actors[No].interactions[a][Adv] = 3;
                    break;
                }
            }
        }
        private bool strategies(int x, int y)
        {
            int strategyNo = actors[x].strategy;
            if (strategyNo == 1) //TFT
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] == ↙
    4)
                        return false;
                    if (actors[x].interactions[a][y] != 0)
                        return true;
                }
                return true;
            }
            else if (strategyNo == 2)  //CHAMPION
            {
                actors[x].encounters[y]++;      // number of occuring interactions including ↙
    present move
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].interactions[a][y] == 3 || actors[x].interactions[a][y]↙
     == 1)
                            actors[x].standing[y]++;
                        if (actors[x].encounters[y] > 10)
                        {
                            if (actors[x].encounters[y] > 25)
```

```
                        {
                            if (actors[x].interactions[a][y] == 2 || actors[x].
interactions[a][y] == 4)
                            {
                                if (actors[x].standing[y] / actors[x].encounters[y] < 0.6
 && actors[x].standing[y] / actors[x].encounters[y] < rnd.NextDouble())
                                    return false;
                            }
                            return true;
                        }
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                            return false;
                    }
                    return true;
                }
            }
            return true;
        }
        else if (strategyNo == 3) //BOERUFSEN
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    int K42R = new int();
                    int ipick = 1;
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y]
 == 2)
                        ipick = 0;                              // works as IPICK in
Axelrod's second tournament
                    int jpick = 1;
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y]
 == 3)
                        jpick = 0;                              // works as JPICK in
Axelrod's second tournament
                    if (actors[x].encounters[y] > 2)
                        actors[x].array[y][(int)actors[x].paramOne[y], jpick]++;
                    if (actors[x].defections[y] != 0)           // works as IDEF in
Axelrod's second tournament
                        K42R = 1;
                    else
                    {
                        if (actors[x].interactions[a][y] == 4)
                        {
                            actors[x].variables[y][10]++;       // works as L3MOV in
Axelrod's second tournament
                            if (actors[x].variables[y][10] < 3)
                                K42R = 1;
                            else
                            {
                                K42R = 0;
                                actors[x].variables[y][10] = 0;
                                actors[x].variables[y][11] = 0; // works as L3ECH in
Axelrod's second tournament
                            }
                        }
                        else
                        {
                            actors[x].variables[y][10] = 0;
                            if (ipick == jpick)
                                actors[x].variables[y][11] = 0;
                            else
                            {
                                if (jpick != actors[x].paramOne[y] || ipick != actors[x].
paramTwo[y])
                                    actors[x].variables[y][11] = 0;
                                else
```

```
                                {
                                    actors[x].variables[y][11]++;
                                    if (actors[x].variables[y][11] >= 3)
                                    {
                                        actors[x].variables[y][11] = 0;
                                        actors[x].variables[y][10] = 0;
                                        actors[x].standing[y] = 1;  // works as ICOOP in ↙
Axelrod's second tournament
                                    }
                                }
                            }
                            K42R = jpick;
                        }
                    }
                    if (actors[x].encounters[y] > 2 && (int)(actors[x].encounters[y] - 2)↙
 % 25 == 0)
                    {
                        actors[x].defections[y] = 0;
                        if (actors[x].array[y][0, 0] + actors[x].array[y][1, 0] <= 17)
                        {
                            if (actors[x].array[y][0, 0] + actors[x].array[y][1, 0] < 3)
                                actors[x].defections[y] = 1;
                            if ((actors[x].array[y][0, 0] + actors[x].array[y][1, 0]) > 7↙
 && (100 * actors[x].array[y][0, 0] / (actors[x].array[y][0, 0] + actors[x].array[y][1,  ↙
0])) < 70)
                                actors[x].defections[y] = 1;
                        }
                        actors[x].array[y][0, 0] = 0;
                        actors[x].array[y][0, 1] = 0;
                        actors[x].array[y][1, 0] = 0;
                        actors[x].array[y][1, 1] = 0;
                        if (actors[x].defections[y] != 0)
                        {
                            actors[x].standing[y] = 0;
                            actors[x].variables[y][11] = 0;
                            actors[x].variables[y][10] = 0;
                            K42R = 1;
                        }
                    }
                    if (actors[x].standing[y] == 1 && K42R == 1)
                    {
                        actors[x].standing[y] = 0;
                        K42R = 0;
                    }
                    actors[x].paramOne[y] = ipick;              // works as I2PCK in     ↙
Axelrod's second tournament
                    actors[x].paramTwo[y] = jpick;              // works as J2PCK in     ↙
Axelrod's second tournament
                    if (K42R == 0)
                        return true;
                    return false;
                }
            }
            actors[x].standing[y] = 0;
            return true;
        }
        else if (strategyNo == 4) //CAVE
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]↙
 == 4)
                        actors[x].defections[y]++;
                    if (actors[x].encounters[y] > 39 && actors[x].defections[y] / actors ↙
[x].encounters[y] > 0.39)
                        return false;
```

```
                    if (actors[x].encounters[y] > 29 && actors[x].defections[y] / actors
[x].encounters[y] > 0.65)
                        return false;
                    if (actors[x].encounters[y] > 19 && actors[x].defections[y] / actors
[x].encounters[y] > 0.79)
                        return false;
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                    {
                        if (actors[x].defections[y] <= 17 && rnd.NextDouble() < 0.5)
                            return true;
                        return false;
                    }
                    return true;
                }
            }
            return true;
        }
        else if (strategyNo == 5) //ADAMS WILLIAM
        {
            actors[x].encounters[y]++;
            if (actors[x].encounters[y] < 3)
            {
                actors[x].standing[y] = 4;      // it works as the number of defections
allowed (AM) in Axelrod's second tournament.
                return true;
            }
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                        actors[x].defections[y]++;
                    break;
                }
            }
            if (actors[x].defections[y] < actors[x].standing[y])
                return true;
            else if (actors[x].defections[y] == actors[x].standing[y])
                return false;
            else
            {
                actors[x].standing[y] = actors[x].standing[y] / 2;
                actors[x].defections[y] = 0;
                if (rnd.NextDouble() < actors[x].standing[y])
                    return true;
                return false;
            }
        }
        else if (strategyNo == 6) //GRAASKAMP & KATZEN
        {
            actors[x].encounters[y]++;
            if (actors[x].encounters[y] == 1)
                actors[x].standing[y] = 0;
            if (actors[x].standing[y] > 0)        // actors[x].standing[y] works as ID in
 Axelrod's code
                return false;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].interactions[a][y] == 1)
                    {
                        if (actors[x].rewardshift[y] - 1 < 0)
                            actors[x].defections[y] += (reward - (reward - punishment) *
(1 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));     // actors[x].
defections[y] works as K in Axelrod's code (score of player x gained from interactions
with actor y so far)
```

```
                        else if (actors[x].rewardshift[y] - 1 >= 0)
                            actors[x].defections[y] += (reward + (temptation - reward) * ↙
(1 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                    }
                    else if (actors[x].interactions[a][y] == 2)
                        actors[x].defections[y] += sucker;
                    else if (actors[x].interactions[a][y] == 3)
                        actors[x].defections[y] += temptation;
                    else
                        actors[x].defections[y] += punishment;
                    if (actors[x].encounters[y] == 11 || actors[x].encounters[y] == 21 ||↙
 actors[x].encounters[y] == 31 || actors[x].encounters[y] == 41 || actors[x].encounters ↙
[y] == 51 || actors[x].encounters[y] == 101)
                    {
                        double score;
                        if (payoffshift == 1)
                            score = (actors[x].encounters[y] - 1) * reward - 7;
                        else
                            score = (actors[x].encounters[y] - 1) * reward + (temptation ↙
- reward) * (actors[x].encounters[y] - 1 - (1 - Math.Pow(payoffshift, actors[x]. ↙
encounters[y] - 1)) / (1 - payoffshift)) - 7;  // finite geometric series
                        if (actors[x].defections[y] < score)
                        {
                            actors[x].standing[y] = 1;
                            return false;
                        }
                    }
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]↙
 == 4)
                        return false;
                    return true;
                }
            }
            return true;
        }
        else if (strategyNo == 7) //WEINER
        {
            actors[x].encounters[y]++;
            int defections = 0;
            int sum = 0;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    for (int b = a - 1; b >= 1; b--)
                    {
                        if (actors[x].interactions[b][y] != 0)
                        {
                            sum++;
                            if (actors[x].interactions[b][y] == 2 || actors[x]. ↙
interactions[b][y] == 4)
                                defections++;       // works as LSUM in Axelrod's second ↙
tournament
                            if (sum == 12)
                                break;
                        }
                    }
                    int q = 0;
                    if (actors[x].standing[y] == 1)
                    {
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions ↙
[a][y] == 4)
                            actors[x].standing[y] = 2;
                    }
                    else if (actors[x].standing[y] == 2)
                    {
                        actors[x].standing[y] = 3;
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions ↙
[a][y] == 4)
```

```
                                actors[x].standing[y] = 1;
                            }
                            else
                            {
                                if (actors[x].defections[y] < actors[x].encounters[y])      //
actors[x].defections[y] works as IFORGV in Ax's second tournament
                                    q = 1;
                                if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                                    actors[x].defections[y] += 20;
                                actors[x].standing[y] = 1;
                            }
                            if (defections < 5)
                            {
                                if (q == 1)
                                    return true;
                                if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                                    return false;
                                return true;
                            }
                            return false;
                        }
                    }
                    return true;
                }
                else if (strategyNo == 8) //HARRINGTON
                {
                    actors[x].encounters[y]++;
                    for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                    {
                        if (actors[x].interactions[a][y] != 0)
                        {
                            int J = 0;                                      // works as J in
Axelrod's second tournament
                            if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                                J = 1;
                            if (actors[x].encounters[y] > 2)
                            {
                                if (actors[x].defections[y] == 1)           // works as IRAND in
Axelrod's second tournament
                                {
                                    actors[x].array[y][0, 0] = actors[x].array[y][0, 0] + 4 * J -
 3;
                                    if (actors[x].array[y][0, 0] >= 11)     // works as IRDCHK in
 Axelrod's second tournament
                                    {
                                        actors[x].defections[y] = 2;
                                        actors[x].array[y][0, 1] = 2;       // works as ICOOP in
Axelrod's second tournamane
                                        return true;
                                    }
                                    return false;
                                }
                                actors[x].array[y][1, 0] += J;              // works as IOPP in
Axelrod's second tournament
                                actors[x].variables[y][(int)actors[x].array[y][1, 1] + 4 * J]++;
    // works as HIST(IND, J+1) in Axelrod's second tournament; VARIABLES[10] - [17]
                                if (actors[x].encounters[y] != 15 && actors[x].encounters[y] % 15
 == 0 && actors[x].defections[y] != 2)
                                {
                                    if (actors[x].variables[y][10] / (actors[x].encounters[y] -
2) < 0.8)
                                    {
                                        if (actors[x].array[y][1, 0] * 4 >= (actors[x].encounters
[y] - 2) && actors[x].array[y][1, 0] * 4 <= (3 * actors[x].encounters[y]) - 6)
                                        {
                                            double[] rows = new double[4];
```

```
                                    for (int q = 0; q < 4; q++)
                                        rows[q] = actors[x].variables[y][10 + q] + actors
[x].variables[y][14 + q];
                                    double[] cols = new double[2];
                                    double sum = 0;
                                    for (int p = 0; p < 2; p++)
                                    {
                                        sum = 0;
                                        for (int w = 0; w < 4; w++)
                                            sum = sum + actors[x].variables[y][10 + w + 4
 * p];
                                        cols[p] = sum;
                                    }
                                    sum = 0;
                                    double ex = new double();
                                    for (int h = 0; h < 4; h++)
                                    {
                                        for (int k = 0; k < 2; k++)
                                        {
                                            ex = rows[h] * cols[k] / (actors[x].
encounters[y] - 2);
                                            if (ex > 1)
                                                sum += Math.Pow(actors[x].variables[y][10
 + h + 4 * k] - ex, 2) / ex;
                                        }
                                    }
                                    if (sum <= 3)
                                    {
                                        actors[x].defections[y] = 1;
                                        return false;
                                    }
                                }
                            }
                        }
                    }
                    if (actors[x].variables[y][18] == 1 && J == 1)  // works as ITRY in
Axelrod's second tournament
                        actors[x].variables[y][19] = 1;             // works as IBURN in
Axelrod's second tournament
                    if (actors[x].encounters[y] <= 37 && J == 0)
                        actors[x].variables[y][20]++;               // works as ITWIN in
Axelrod's second tournament
                    if (actors[x].encounters[y] == 38 && J == 1)
                        actors[x].variables[y][20]++;
                    if (actors[x].encounters[y] >= 39 && actors[x].variables[y][20] == 37
 && J == 1)
                        actors[x].variables[y][20] = 0;
                    int decision = new int();
                    if (actors[x].variables[y][20] == 37)
                    {
                        decision = 0;
                        actors[x].variables[y][18]--;
                        actors[x].array[y][0, 1]--;
                    }
                    else
                    {
                        actors[x].variables[y][21] = actors[x].variables[y][21] * J + J;
   // works as IDEF in Axelrod's second tournament
                        if (actors[x].variables[y][21] >= 20)
                        {
                            actors[x].variables[y][21 + (int)actors[x].standing[y]]++;
        // works as ID(IPARTY) in Axelrod's second tournament;
                            decision = 1;
                        }
                        else
                        {
                            actors[x].standing[y] = 3 - actors[x].standing[y];
   // works as IPARTY in Axelrod's second tournament
                            actors[x].variables[y][21 + (int)actors[x].standing[y]] =
```

```
actors[x].variables[y][21 + (int)actors[x].standing[y]] * J + J;
                            if (actors[x].variables[y][21 + (int)actors[x].standing[y]] >
= actors[x].paramOne[y] || actors[x].array[y][0, 1] >= 1)
                            {
                                if (actors[x].variables[y][21 + (int)actors[x].standing
[y]] >= actors[x].paramOne[y])
                                {
                                    actors[x].variables[y][21 + (int)actors[x].standing
[y]] = 0;
                                    actors[x].variables[y][24]++;   // works as ISTRNG in
 Axelrod's second tournament
                                    if (actors[x].variables[y][24] == 8)
                                        actors[x].paramOne[y] = 3;
                                }
                                decision = 0;
                                actors[x].variables[y][18]--;
                                actors[x].array[y][0, 1]--;
                            }
                            else
                            {
                                if (actors[x].encounters[y] >= 37 && actors[x].variables
[y][19] != 1)
                                {
                                    if (actors[x].encounters[y] == 37 || rnd.NextDouble()
 <= actors[x].paramTwo[y])
                                    {
                                        actors[x].variables[y][18] = 2;
                                        actors[x].array[y][0, 1] = 2;
                                        actors[x].paramTwo[y] += 0.05;
                                        decision = 1;
                                    }
                                    else
                                    {
                                        if (J == 0)
                                        {
                                            decision = 0;
                                            actors[x].variables[y][18]--;
                                            actors[x].array[y][0, 1]--;
                                        }
                                        else
                                        {
                                            decision = 1;
                                            actors[x].variables[y][21 + (int)actors[x].
standing[y]]++;
                                        }
                                    }
                                }
                                else
                                {
                                    if (J == 0)
                                    {
                                        decision = 0;
                                        actors[x].variables[y][18]--;
                                        actors[x].array[y][0, 1]--;
                                    }
                                    else
                                    {
                                        decision = 1;
                                        actors[x].variables[y][21 + (int)actors[x].
standing[y]]++;
                                    }
                                }
                            }
                        }
                    }
                    actors[x].array[y][1, 1] = 9 + J + 1;           // works as IND in
Axelrod's second tournament
                    if (actors[x].interactions[a][y] == 3 || actors[x].interactions[a][y]
 == 4)
```

```
                        actors[x].array[y][1, 1] = 9 + 3 + J;
                    if (decision == 0)
                        return true;
                    return false;
                }
            }
            actors[x].paramOne[y] = 5;                              // works as INDEF in ↙
Axelrod's second tournament
            actors[x].paramTwo[y] = 0.2;                            // works as PROB in  ↙
Axelrod's second tournament
            return true;
        }
        else if (strategyNo == 9) //TIDEMAN & CHIERUZZI
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    actors[x].array[y][0, 1] = 1;                   // works as K84R in  ↙
Axelrod's second tournament
                    if (actors[x].interactions[a][y] == 1)
                    {
                        if (actors[x].rewardshift[y] - 1 < 0)
                        {
                            actors[x].paramOne[y] += (reward - (reward - punishment) * (1↙
 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));
                            actors[x].paramTwo[y] += (reward - (reward - punishment) * (1↙
 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));
                        }
                        else if (actors[x].rewardshift[y] - 1 >= 0)
                        {
                            actors[x].paramOne[y] += (reward + (temptation - reward) * (1↙
 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                            actors[x].paramTwo[y] += (reward + (temptation - reward) * (1↙
 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                        }
                    }
                    else if (actors[x].interactions[a][y] == 2)
                    {
                        actors[x].paramOne[y] += sucker;
                        actors[x].paramTwo[y] += temptation;
                        actors[x].defections[y]++;
                    }
                    else if (actors[x].interactions[a][y] == 3)
                    {
                        actors[x].paramOne[y] += temptation;
                        actors[x].paramTwo[y] += sucker;
                    }
                    else
                    {
                        actors[x].paramOne[y] += punishment;            // works as K in ↙
Axelrod's second tournament (IS)
                        actors[x].paramTwo[y] += punishment;            // works as L in ↙
Axelrod's second tournament (JS)
                        actors[x].defections[y]++;                      // works as FJD  ↙
in Axelrod's second tournament
                    }
                    if (actors[x].array[y][0, 0] == 1)                  // works as ISIG ↙
in Axelrod's second tournament
                    {
                        actors[x].array[y][0, 0] = actors[x].encounters[y];
                        actors[x].standing[y] = 0;                      // works as JQ in↙
 Axelrod's second tournament
                        actors[x].array[y][1, 0] = 0;                   // works as JDR  ↙
in Axelrod's second tournament
                        actors[x].array[y][1, 1] = actors[x].paramOne[y] - actors[x].    ↙
paramTwo[y];       // works as DS in Axelrod's second tournament
                        return true;
```

```
                    }
                    if (actors[x].standing[y] == 0 && (actors[x].interactions[a][y] == 2 ↙
|| actors[x].interactions[a][y] == 4))
                        actors[x].array[y][1, 0] += 1;
                    if (actors[x].paramOne[y] - actors[x].paramTwo[y] - actors[x].array ↙
[y][1, 1] - (int)(5 * actors[x].array[y][1, 0] * (actors[x].array[y][1, 0] - 1) / 2) >= ↙
0)
                        actors[x].array[y][0, 1] = 0;
                    if (actors[x].array[y][0, 1] == 1)
                    {
                        if ((actors[x].standing[y] == 0 && (actors[x].interactions[a][y] ↙
== 2 || actors[x].interactions[a][y] == 4)) || (actors[x].encounters[y] - actors[x].array↙
[y][0, 0] < 10))
                        {
                            actors[x].standing[y] = 0;
                            if (actors[x].interactions[a][y] == 2 || actors[x]. ↙
interactions[a][y] == 4)
                                actors[x].standing[y] = 1;
                            return false;
                        }
                        actors[x].standing[y] = 0;
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions ↙
[a][y] == 4)
                            actors[x].standing[y] = 1;
                        if (Math.Abs(actors[x].defections[y] - (actors[x].encounters[y] -↙
 1) / 2) < (1.5 * Math.Sqrt(actors[x].encounters[y] - 1)))
                            return false;
                        actors[x].array[y][0, 0] = 1;
                        return true;
                    }
                    actors[x].standing[y] = 0;
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]↙
 == 4)
                        actors[x].standing[y] = 1;
                    return true;
                }
            }
            actors[x].standing[y] = 0;
            return true;
        }
        else if (strategyNo == 10) //KLUEPFEL
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    actors[x].paramTwo[y] = actors[x].paramOne[y];      // works as J2 ↙
and J1 in Axelrod's second tournament, respectively
                    actors[x].paramOne[y] = actors[x].variables[y][11];
                    actors[x].standing[y] = actors[x].variables[y][10]; // works as I2 in↙
 Axelrod's second tournament
                    actors[x].variables[y][11] = 0;
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]↙
 == 4)
                        actors[x].variables[y][11] = 1;                 // works as J in ↙
Axelrod's second tournament
                    actors[x].variables[y][10] = 0;
                    if (actors[x].interactions[a][y] == 3 || actors[x].interactions[a][y]↙
 == 4)
                        actors[x].variables[y][10] = 1;                 // works as JA in↙
 Axelrod's second tournament; or also as I1
                    if (actors[x].encounters[y] > 2)
                    {
                        if (actors[x].standing[y] == 0)
                        {
                            if (actors[x].variables[y][11] == 0)
                                actors[x].array[y][1, 0]++;             // works as C3 in↙
 Axelrod's second tournament, i.e. opponent's cooperation after my cooperation
```

```
                                else
                                    actors[x].array[y][1, 1]++;                 // works as
C4 in Axelrod's second tournament, i.e. opponent's defection after my cooperation
                            }
                            else
                            {
                                if (actors[x].variables[y][11] == 0)
                                    actors[x].array[y][0, 0]++;             // works as C1 in
 Axelrod's second tournament, i.e. opponent's cooperation after my defection
                                else
                                    actors[x].array[y][0, 1]++;                 // works as
C2 in Axelrod's second tournament, i.e. opponent's defection after my defection
                            }
                            if (actors[x].encounters[y] >= 27)
                            {
                                if (actors[x].array[y][0, 0] >= (actors[x].array[y][0, 0] +
actors[x].array[y][0, 1] - 1.5 * Math.Sqrt(actors[x].array[y][0, 0] + actors[x].array[y]
[0, 1])) / 2 && actors[x].array[y][1, 1] >= (actors[x].array[y][1, 0] + actors[x].array
[y][1, 1] - 1.5 * Math.Sqrt(actors[x].array[y][1, 0] + actors[x].array[y][1, 1])) / 2)
                                    return false;
                            }
                        }
                        if (actors[x].paramOne[y] != actors[x].variables[y][11])
                        {
                            actors[x].defections[y] = 0.6;                  // works as P in
Axelrod's second tournament
                            if (actors[x].variables[y][11] == 0)
                                actors[x].defections[y] = 0.7;
                            if (rnd.NextDouble() >= actors[x].defections[y])
                            {
                                if (actors[x].variables[y][11] == 0)
                                    return false;
                                return true;
                            }
                        }
                        else
                        {
                            if (actors[x].paramTwo[y] != actors[x].paramOne[y])
                            {
                                if (rnd.NextDouble() >= 0.9)
                                {
                                    if (actors[x].variables[y][11] == 0)
                                        return false;
                                    return true;
                                }
                            }
                        }
                        if (actors[x].variables[y][11] == 0)
                            return true;
                        return false;
                    }
                }
                actors[x].standing[y] = 0;                                  // works as I2 in
 Axelrod's second tournament, i.e. my second previous choice
                return true;
            }
            else if (strategyNo == 11) //GETZLER - basically a forgeting function. possible
to work even with rounds, not only with interactions [less often interacting players
forget faster].
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        double last = 0;
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                            last = 1;
                        actors[x].encounters[y] = (actors[x].encounters[y] + last) * 0.5;
```

```
                    break;
                }
            }
            if (actors[x].encounters[y] > rnd.NextDouble())
                return false;
            return true;
        }
        else if (strategyNo == 12) //LEYVRAZ
        {
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    actors[x].defections[y] = actors[x].encounters[y];          // actors
[x].defections[y] works as J2 in Axelrod's tournament
                    actors[x].encounters[y] = actors[x].standing[y];
                    actors[x].standing[y] = 0;
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                        actors[x].standing[y] = 1;
                    if (actors[x].encounters[y] * actors[x].standing[y] == 1)   // actors
[x].encounters[y] works as J1 in Axelrod's tournament
                    {
                        if (rnd.NextDouble() < 0.75)
                            return false;
                        return true;
                    }
                    if (actors[x].defections[y] * 2 + actors[x].encounters[y] + actors[x]
.standing[y] * 2 == 1) // actors[x].defections[y] works as J2 in Axelrod's tournament
                        return false;
                    if (actors[x].defections[y] * 2 + actors[x].encounters[y] * 2 +
actors[x].standing[y] == 1)
                    {
                        if (rnd.NextDouble() < 0.5)
                            return false;
                        return true;
                    }
                    return true;
                }
            }
            actors[x].standing[y] = 0;
            return true;
        }
        else if (strategyNo == 13) //WHITE
        {
            double N = 1;
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                    {
                        actors[x].defections[y]++;
                        if (actors[x].encounters[y] > 10)
                            N = (int)(Math.Log(actors[x].encounters[y]));   // ln(11) = 2
; ln(10000) = 9
                    }
                    if (rnd.NextDouble() <= (int)((N * actors[x].defections[y]) / actors
[x].encounters[y]))     // 6 defections of 11 encounters; or 1112 defections of 10000
encounters needed for replying with defection upon defection
                        return false;
                    return true;
                }
            }
            return true;
        }
        else if (strategyNo == 14) //EATHERLEY
```

```
            {
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] ==
4)
                    {
                        actors[x].defections[y]++;
                        if (actors[x].defections[y] / (actors[x].encounters[y] - 1) > rnd.
NextDouble())
                            return false;
                        return true;
                    }
                    if (actors[x].interactions[a][y] != 0)
                        return true;
                }
                return true;
            }
            else if (strategyNo == 15) //BLACK
            {
                actors[x].encounters[y]++;
                actors[x].standing[y] = 0;
                actors[x].defections[y] = 0;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        actors[x].standing[y]++;
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                            actors[x].defections[y]++;
                        if (actors[x].standing[y] == 5)
                        {
                            if (rnd.NextDouble() * 25 < actors[x].defections[y] * actors[x].
defections[y] - 1)
                                return false;
                            return true;
                        }
                    }
                }
                return true;
            }
            else if (strategyNo == 16) //HUFFORD RICHARD
            {
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].encounters[y] <= actors[x].defections[y] && (actors[x].
interactions[a][y] == 2 || actors[x].interactions[a][y] == 4))
                            actors[x].defections[y] = actors[x].encounters[y] + (int)((20 *
actors[x].array[y][1, 0]) / actors[x].array[y][1, 1]) + 1;
                        actors[x].standing[y] = actors[x].standing[y] % 4 + 1;        //
works as N in Axelrod's second tournament
                        actors[x].paramTwo[y] -= actors[x].variables[y][9 + (int)actors[x].
standing[y]];
                        actors[x].variables[y][9 + (int)actors[x].standing[y]] = 0;
                        if ((actors[x].array[y][0, 0] == 0 && (actors[x].interactions[a][y] =
= 1 || actors[x].interactions[a][y] == 3)) || (actors[x].array[y][0, 0] == 1 && (actors
[x].interactions[a][y] == 2 || actors[x].interactions[a][y] == 4)))
                        {
                            actors[x].paramOne[y] += 1;
                            actors[x].paramTwo[y] += 1;
                            actors[x].variables[y][9 + (int)actors[x].standing[y]] = 1;
                        }
                        actors[x].array[y][0, 0] = 0;   // works as MYLAST in Axelrod's
second tournament
                        if (actors[x].interactions[a][y] == 3 || actors[x].interactions[a][y]
```

```
 == 4)
                        actors[x].array[y][0, 0] = 1;
                    if (actors[x].encounters[y] < actors[x].defections[y] + 2)
                    {
                        if (actors[x].encounters[y] == actors[x].defections[y])
                            return false;
                        if (actors[x].paramOne[y] < 0.625 * actors[x].encounters[y] ||
actors[x].paramTwo[y] < 3)
                            return false;
                        if (actors[x].paramOne[y] > 0.9 * actors[x].encounters[y] &&
actors[x].paramTwo[y] == 5)
                            return true;
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                            return false;
                        return true;
                    }
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                        actors[x].array[y][1, 0] += 1;
                    else
                        actors[x].array[y][1, 1] += 1;
                    actors[x].defections[y] = actors[x].encounters[y] + (int)((20 *
actors[x].array[y][1, 0]) / actors[x].array[y][1, 1]) + 1;
                    return true;
                }
            }
            actors[x].array[y][1, 0] = 2;           // works as NUM in Axelrod's second
tournament
            actors[x].array[y][1, 1] = 2;           // works as DEN in Axelrod's second
tournament
            actors[x].defections[y] = 20;           // works as RF in Axelrod's second
tournament
            actors[x].paramOne[y] = 2;              // works as LONG in Axelrod's second
tournament
            actors[x].paramTwo[y] = 5;              // works as SHORT in Axelrod's second
 tournament
            actors[x].standing[y] = 2;
            for (int q = 0; q < 5; q++)
                actors[x].variables[y][10 + q] = 1;     // works as SH2(N) in Axelrod's
second tournament, [10]-[14]
            return true;
        }
        else if (strategyNo == 17) //YAMACHI
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].array[y][(int)actors[x].paramOne[y], (int)actors[x].
paramTwo[y]] >= 0)
                        actors[x].standing[y] = 1;
                    else
                        actors[x].standing[y] = 0;
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                    {
                        actors[x].defections[y]++;      // works as F in Axelrod's second
 tournament
                        actors[x].array[y][(int)actors[x].paramOne[y], (int)actors[x].
paramTwo[y]] -= 1;
                        actors[x].paramOne[y] = 1;      // works as X in Axelrod's second
 tournament
                    }
                    else
                    {
                        actors[x].array[y][(int)actors[x].paramOne[y], (int)actors[x].
paramTwo[y]] += 1;
```

```
                        actors[x].paramOne[y] = 0;
                    }
                    actors[x].paramTwo[y] = 1 - actors[x].standing[y];      // works as Y
 in Axelrod's second tournament
                    if (actors[x].encounters[y] > 40 && (10 * Math.Abs(actors[x].
encounters[y] - 1 - 2 * actors[x].defections[y]) < actors[x].encounters[y]))
                        return false;
                    else
                    {
                        if (actors[x].standing[y] == 1)
                            return true;
                        return false;
                    }
                }
            }
            return true;
        }
        else if (strategyNo == 18) //COLBERT
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].encounters[y] < 9)
                    {
                        actors[x].standing[y] = 0;                  // works as LASTI in
Axelrod's second tournament
                        if (actors[x].encounters[y] == 6)
                            return false;
                        return true;
                    }
                    actors[x].standing[y]--;
                    if (actors[x].standing[y] > 0)
                    {
                        if (actors[x].standing[y] == 3)
                            return false;
                        return true;
                    }
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                    {
                        actors[x].standing[y] = 4;
                        return false;
                    }
                    return true;
                }
            }
            return true;
        }
        else if (strategyNo == 19) //MAUK
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].interactions[a][y] == 1)
                    {
                        if (actors[x].rewardshift[y] - 1 < 0)
                            actors[x].paramOne[y] += (reward - (reward - punishment) * (1
 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));
                        else
                            actors[x].paramOne[y] += (reward + (temptation - reward) * (1
 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                    }
                    else if (actors[x].interactions[a][y] == 2)
                        actors[x].paramOne[y] += temptation;
                    else if (actors[x].interactions[a][y] == 3)
```

```
                        actors[x].paramOne[y] += sucker;
                    else
                        actors[x].paramOne[y] += punishment;    // works as JSCOR in
Axelrod's second tournament
                    if (actors[x].encounters[y] < 51)
                    {
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                            return false;
                        return true;
                    }
                    if (actors[x].encounters[y] == 51)
                        return false;
                    if (actors[x].encounters[y] < 57)
                    {
                        if (actors[x].encounters[y] > 52)
                        {
                            actors[x].variables[y][(int)actors[x].encounters[y] - 43] =
actors[x].defections[y];
                            if (actors[x].interactions[a][y] == 2 || actors[x].
interactions[a][y] == 4)
                                actors[x].variables[y][(int)actors[x].encounters[y] - 43]
 = actors[x].defections[y] + 1;
                        }                                       // VARIABLES[10-13] work
as NMOV in Axelrod's second tournament
                        actors[x].defections[y] = 4;            // works as MMOVE in
Axelrod's second tournament
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                            return false;
                        actors[x].defections[y] = 2;
                        return true;
                    }
                    if (actors[x].encounters[y] == 57)
                    {
                        if (actors[x].paramOne[y] > 135)
                        {
                            if (actors[x].variables[y][11] == 3 && actors[x].variables[y]
[10] == 2 && actors[x].variables[y][12] == 4 && actors[x].variables[y][13] == 2)
                            {
                                actors[x].standing[y] = 4;
                                return true;
                            }
                            else if (actors[x].variables[y][11] == 4 && actors[x].
variables[y][10] == 3 && actors[x].variables[y][12] == 3)
                            {
                                actors[x].standing[y] = 2;
                                return true;
                            }
                            else if (actors[x].variables[y][11] == 5 && actors[x].
variables[y][10] == 5 && actors[x].variables[y][12] == 5)
                            {
                                actors[x].standing[y] = 2;
                                return true;
                            }
                            actors[x].standing[y] = 1;
                            actors[x].paramTwo[y] = (int)(rnd.NextDouble() * 10 + 5);
  // works as N in Axelrod's second tournament
                            return true;
                        }
                        actors[x].standing[y] = 3;              // works as IGAME in
Axelrod's second tournament
                        return false;
                    }
                    if (actors[x].standing[y] == 1)
                    {
                        if (actors[x].paramTwo[y] > 0)
                        {
                            actors[x].paramTwo[y]--;
```

```
                                if (actors[x].interactions[a][y] == 2 || actors[x].
interactions[a][y] == 4)
                                    return false;
                                return true;
                            }
                            actors[x].paramTwo[y] = (int)(rnd.NextDouble() * 10 + 5);
                            return false;
                        }
                        else if (actors[x].standing[y] == 2)
                        {
                            if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                                return false;
                            return true;
                        }
                        else if (actors[x].standing[y] == 3)
                            return false;
                        if (actors[x].encounters[y] >= 118)
                            actors[x].standing[y] = 2;
                        return true;
                    }
                }
                return true;
            }
            else if (strategyNo == 20) //MIKKELSON
            {
                actors[x].encounters[y]++;
                if (actors[x].encounters[y] == 1)
                    actors[x].standing[y] = -3;       // works as J2 in Axelrod's code
                int choice = 0;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                        {
                            actors[x].defections[y]++;
                            choice = 1;
                        }
                        break;
                    }
                }
                actors[x].standing[y] = actors[x].standing[y] - 1 + 3 * choice;
                if (actors[x].standing[y] > 10)
                    actors[x].standing[y] = 10;
                if (actors[x].standing[y] < -5)
                    actors[x].standing[y] = -5;
                if (actors[x].encounters[y] < 3)
                    return true;
                if (actors[x].standing[y] < 3)
                    return true;
                if (actors[x].encounters[y] > 10)
                {
                    if (actors[x].defections[y] / actors[x].encounters[y] < 0.15)
                        return true;
                }
                else
                    actors[x].standing[y] = -1;
                return false;
            }
            else if (strategyNo == 21) //ROWSAM
            {
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].interactions[a][y] == 1)
```

```
                    {
                        if (actors[x].rewardshift[y] - 1 < 0)
                            actors[x].defections[y] += (reward - (reward - punishment) * ↙
(1 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));     // actors[x]. ↙
defections[y] works as K in Axelrod's code (score of player x gained from interactions ↙
with actor y so far)
                        else if (actors[x].rewardshift[y] - 1 >= 0)
                            actors[x].defections[y] += (reward + (temptation - reward) * ↙
(1 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                    }
                    else if (actors[x].interactions[a][y] == 2)
                        actors[x].defections[y] += sucker;
                    else if (actors[x].interactions[a][y] == 3)
                        actors[x].defections[y] += temptation;
                    else
                        actors[x].defections[y] += punishment;          // works as K in ↙
Axelrod's second tournament
                    break;
                }
            }
            if (actors[x].paramOne[y] > 6)          // works as KAM in Axelrod's second ↙
tournament
                return false;
            if (actors[x].paramTwo[y] >= 1)         // works as NPHA in Axelrod's second ↙
tournament
            {
                actors[x].paramTwo[y] -= 1;
                if (actors[x].paramTwo[y] == 0)
                    return false;
                return true;
            }
            if ((int)actors[x].encounters[y] % 18 == 0 && actors[x].paramOne[y] > 2)
                actors[x].paramOne[y] -= 1;
            if ((int)actors[x].encounters[y] % 6 != 0)
                return true;
            if (actors[x].defections[y] >= actors[x].encounters[y] * (punishment + 0.75 *↙
 (reward - punishment)))
                return true;
            actors[x].paramOne[y] += 1;
            if (actors[x].defections[y] < actors[x].encounters[y] * (punishment + 0.5 *  ↙
(reward - punishment)))
            {
                actors[x].paramOne[y] += 1;
                if (actors[x].defections[y] < actors[x].encounters[y] * (punishment + 0. ↙
25 * (reward - punishment)))
                {
                    actors[x].paramOne[y] += 1;
                    if (actors[x].defections[y] < actors[x].encounters[y] * punishment)
                        actors[x].paramOne[y] += 2;
                }
            }
            actors[x].paramTwo[y] = 2;
            return false;
        }
        else if (strategyNo == 22) //APPOLD
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    actors[x].paramTwo[y] = actors[x].paramOne[y];
                    actors[x].paramOne[y] = 1;
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y]↙
 == 2)
                        actors[x].paramOne[y] = 0;          // works as LMV in Axelrod's ↙
second tournament
                    if (actors[x].paramTwo[y] == 0)         // works as MMV in Axelrod's ↙
second tournament
```

```
                    {
                        actors[x].defections[y]++;          // works as MMC in Axelrod's ↙
second tournament
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions  ↙
[a][y] == 4)
                            actors[x].array[y][0, 0]++;     // works as MP in Axelrod's  ↙
second tournament
                    }
                    else
                    {
                        actors[x].standing[y]++;            // works as MMD in Axelrod's ↙
second tournament
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions  ↙
[a][y] == 4)
                            actors[x].array[y][1, 1]++;     // works as MP2 in Axelrod's ↙
second tournament
                    }
                    if (actors[x].encounters[y] <= 4)
                        return true;
                    if ((actors[x].interactions[a][y] == 2 || actors[x].interactions[a]  ↙
[y] == 4) && actors[x].array[y][1, 0] == 0)
                    {
                        actors[x].array[y][1, 0] = 1;       // works as DFLG in Axelrod's↙
 second tournament
                        return true;
                    }
                    if (actors[x].paramTwo[y] == 0 && rnd.NextDouble() < actors[x].array ↙
[y][0, 0] / actors[x].defections[y])
                        return false;
                    if (actors[x].paramTwo[y] == 1 && rnd.NextDouble() < actors[x].array ↙
[y][1, 1] / actors[x].standing[y])
                        return false;
                    return true;
                }
            }
            return true;
        }
        else if (strategyNo == 23) //GRISELL
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] == ↙
4)
                    actors[x].defections[y]++;
                if (actors[x].interactions[a][y] != 0)
                    break;
            }
            if (actors[x].defections[y] / actors[x].encounters[y] < 0.5)
                return true;
            return false;
        }
        else if (strategyNo == 24) //TF2T & MAYNARD SMITH
        {
            for (int a = actors[x].interactions.Count - 1; a >= 2; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].standing[y] == 0 && (actors[x].interactions[a][y] == 2 ↙
|| actors[x].interactions[a][y] == 4))
                        return false;
                    actors[x].standing[y] = 1;          // works as penultimate opponent ↙
's move
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]↙
 == 4)
                        actors[x].standing[y] = 0;
                    return true;
                }
            }
```

```
                return true;
            }
            else if (strategyNo == 25) //ALMY
            {
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        actors[x].variables[y][15]--;       // works as COUNT in Axelrod's
second tournament
                        bool cont = new bool();
                        double oldstep = new double();
                        double voldmv = actors[x].variables[y][11];
                        if (actors[x].interactions[a][y] == 1)
                        {
                            if (actors[x].rewardshift[y] - 1 < 0)
                                actors[x].variables[y][16] += (reward - (reward - punishment)
 * (1 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));     // actors[x]
.defections[y] works as K in Axelrod's code (score of player x gained from interactions
with actor y so far)
                            else if (actors[x].rewardshift[y] - 1 >= 0)
                                actors[x].variables[y][16] += (reward + (temptation - reward)
 * (1 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                        }
                        else if (actors[x].interactions[a][y] == 2)
                            actors[x].variables[y][16] += sucker;
                        else if (actors[x].interactions[a][y] == 3)
                            actors[x].variables[y][16] += temptation;
                        else
                            actors[x].variables[y][16] += punishment;          // works as K
in Axelrod's second tournament
                        if (actors[x].interactions[a][y] == 4)
                            actors[x].array[y][0, 1]++;     // works as BOTHD in Axelrod's
second tournament
                        else
                            actors[x].array[y][0, 1] = 0;
                        actors[x].variables[y][11] = 0;     // works as OLDMOV in Axelrod's
second tournament
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                        {
                            actors[x].variables[y][11] = 1;
                            actors[x].array[y][1, 1]++;     // works as TATCNT in Axelrod's
second tournament
                            if (actors[x].array[y][1, 0] == 0)  // works as EVIL in Axelrod's
 second tournament
                                actors[x].array[y][1, 0] = 1;
                        }
                        do
                        {
                            cont = true;
                            if (actors[x].paramOne[y] != 5 && actors[x].paramOne[y] != 4 &&
actors[x].paramTwo[y] == 1)
                            {
                                actors[x].variables[y][15] = 10;
                                actors[x].array[y][0, 0] = 0;
                                actors[x].array[y][1, 1] = 0;
                                actors[x].paramTwo[y] = 2;
                                continue;
                            }
                            if (actors[x].paramOne[y] != 5 && actors[x].paramOne[y] != 4 &&
actors[x].paramTwo[y] == 3)
                            {
                                oldstep = actors[x].paramOne[y];
                                actors[x].variables[y][11 + (int)actors[x].paramOne[y]] =
actors[x].variables[y][16] - actors[x].variables[y][10];  // works as OK() in Axelrod's
second tournament
                                actors[x].variables[y][10] = actors[x].variables[y][16];  //
```

```
    works as TOTK in Axelrod's second tournament
                                actors[x].paramTwo[y] = 1;
                                if (actors[x].array[y][1, 1] > 0)
                                {
                                    actors[x].paramOne[y] = 1;
                                    for (int s = 0; s < 2; s++)
                                    {
                                        for (int t = 0; t < 2; t++)
                                        {
                                            if (actors[x].variables[y][12 + s] != 0 && actors↙
    [x].variables[y][13 + t] != 0)
                                            {
                                                if (actors[x].variables[y][12 + s] < actors   ↙
    [x].variables[y][13 + t])
                                                {
                                                    if (actors[x].paramOne[y] == s + 1)
                                                        actors[x].paramOne[y] = t + 2;
                                                }
                                            }
                                        }
                                    }
                                    if (actors[x].paramOne[y] != 3 && actors[x].variables[y] ↙
    [11 + (int)actors[x].paramOne[y] + 1] == 0 && (actors[x].array[y][1, 1] >= 4 || actors[x]↙
    .array[y][0, 0] == 0))
                                        actors[x].paramOne[y]++;
                                    if (actors[x].paramOne[y] < oldstep && actors[x].array[y]↙
    [0, 1] > 0)
                                        actors[x].paramOne[y] = 5;
                                    continue;
                                }
                                actors[x].paramOne[y] = 4;
                                if (actors[x].array[y][1, 0] == 0)
                                    actors[x].paramOne[y] = 1;
                                if (actors[x].array[y][1, 0] == 0)
                                    actors[x].array[y][1, 0] = -1;
                                continue;
                            }
                            if (actors[x].paramOne[y] == 1)
                            {
                                if (actors[x].variables[y][15] == 0)
                                    actors[x].paramTwo[y] = 3;
                                if (voldmv + actors[x].variables[y][11] == 2)
                                {
                                    actors[x].array[y][0, 0]++;
                                    return false;
                                }
                                return true;
                            }
                            if (actors[x].paramOne[y] == 2)
                            {
                                if (actors[x].variables[y][15] == 0)
                                    actors[x].paramTwo[y] = 3;
                                if (actors[x].variables[y][11] == 1)
                                {
                                    actors[x].array[y][0, 0]++;
                                    return false;
                                }
                                return true;
                            }
                            if (actors[x].paramOne[y] == 3)
                            {
                                if (actors[x].variables[y][15] == 0)
                                    actors[x].paramTwo[y] = 3;
                                actors[x].array[y][0, 0]++;
                                return false;
                            }
                            if (actors[x].paramOne[y] == 4)
                            {
                                if (actors[x].paramTwo[y] == 1)
```

```
                                {
                                    actors[x].paramTwo[y] = 2;
                                    actors[x].variables[y][15] = actors[x].standing[y];
                                    actors[x].array[y][1, 1] = 0;
                                    return false;
                                }
                                if (actors[x].paramTwo[y] == 2)
                                {
                                    if (actors[x].variables[y][15] == 0)
                                        actors[x].paramTwo[y] = 3;
                                    return true;
                                }
                                if (actors[x].paramTwo[y] == 3)
                                {
                                    if (actors[x].array[y][1, 1] == 0)
                                    {
                                        actors[x].paramTwo[y] = 2;
                                        actors[x].defections[y] = 1;    // works as F in
Axelrod's second tournament
                                        actors[x].variables[y][15] = actors[x].standing[y];
                                        return false;
                                    }
                                    if (actors[x].defections[y] == 0)
                                    {
                                        actors[x].paramTwo[y] = 4;
                                        if (actors[x].variables[y][11] == 1)
                                            actors[x].standing[y]++;
                                        actors[x].array[y][1, 1] = actors[x].variables[y][11]
;
                                        return true;
                                    }
                                    actors[x].standing[y]++;
                                    actors[x].paramTwo[y] = 1;
                                    actors[x].paramOne[y] = 1;
                                    continue;
                                }
                                if (actors[x].paramTwo[y] == 4)
                                {
                                    if (actors[x].array[y][1, 1] <= 4)
                                        return true;
                                    actors[x].paramTwo[y] = 1;
                                    actors[x].paramOne[y] = 1;
                                    continue;
                                }
                            }
                            if (actors[x].paramOne[y] == 5)
                            {
                                if (actors[x].paramTwo[y] != 2)
                                {
                                    actors[x].variables[y][15] = 5;
                                    actors[x].paramTwo[y] = 2;
                                }
                                if (actors[x].variables[y][15] != 0)
                                    return true;
                                actors[x].paramTwo[y] = 1;
                                actors[x].paramOne[y] = 1;
                                for (int s = 0; s < 2; s++)
                                {
                                    for (int t = 0; t < 2; t++)
                                    {
                                        if (actors[x].variables[y][12 + s] != 0 && actors[x].
variables[y][13 + t] != 0)
                                        {
                                            if (actors[x].variables[y][12 + s] < actors[x].
variables[y][13 + t])
                                            {
                                                if (actors[x].paramOne[y] == s + 1)
                                                    actors[x].paramOne[y] = t + 2;
                                            }
```

```
                                    }
                                }
                            }
                            if (actors[x].paramOne[y] != 3 && actors[x].variables[y][11 +
 (int)actors[x].paramOne[y] + 1] == 0 && (actors[x].array[y][1, 1] >= 4 || actors[x].
array[y][0, 0] == 0))
                                actors[x].paramOne[y]++;
                            if (actors[x].paramOne[y] < oldstep && actors[x].array[y][0,
1] > 0)
                                actors[x].paramOne[y] = 5;
                            continue;
                        }
                    }
                    while (cont == true);
                }
            }
            actors[x].paramOne[y] = 1;      // works as STEP in Axelrod's second
tournament
            actors[x].paramTwo[y] = 2;      // works as SUBSTP in Axelrod's second
tournament
            actors[x].variables[y][15] = 10;
            return true;
        }
        else if (strategyNo == 26) //AMBUEHL & HICKEY
        {
            actors[x].encounters[y]++;
            actors[x].standing[y] = 0;
            actors[x].defections[y] = 0;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].encounters[y] > 5)
                    {
                        for (int b = a - 1; b >= 1; b--)
                        {
                            if (actors[x].interactions[b][y] != 0)
                            {
                                actors[x].standing[y]++;
                                if (actors[x].interactions[b][y] == 2 || actors[x].
interactions[b][y] == 4)
                                    actors[x].defections[y]++;
                                if (actors[x].standing[y] == 5)
                                    break;
                            }
                        }
                        if (actors[x].defections[y] < 3)
                            return true;
                        return false;
                    }
                    else
                    {
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                            return false;
                        return true;
                    }
                }
            }
            return true;
        }
        else if (strategyNo == 27) //FEATHERS
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].interactions[a][y] == 1)
```

```
                    {
                        if (actors[x].rewardshift[y] - 1 < 0)
                        {
                            actors[x].paramOne[y] += (reward - (reward - punishment) * (1 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));
                            actors[x].paramTwo[y] += (reward - (reward - punishment) * (1 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));
                            actors[x].variables[y][13] = reward - (reward - punishment) * (1 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1)));
                        }
                        else
                        {
                            actors[x].paramOne[y] += (reward + (temptation - reward) * (1 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                            actors[x].paramTwo[y] += (reward + (temptation - reward) * (1 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                            actors[x].variables[y][13] = reward + (temptation - reward) * (1 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1));
                        }
                    }
                    else if (actors[x].interactions[a][y] == 2)
                    {
                        actors[x].paramOne[y] += sucker;
                        actors[x].paramTwo[y] += temptation;
                        actors[x].variables[y][13] = sucker;
                    }
                    else if (actors[x].interactions[a][y] == 3)
                    {
                        actors[x].paramOne[y] += temptation;
                        actors[x].paramTwo[y] += sucker;
                        actors[x].variables[y][13] = temptation;
                    }
                    else
                    {
                        actors[x].paramOne[y] += punishment;            // works as K in Axelrod's second tournament
                        actors[x].paramTwo[y] += punishment;            // works as L in Axelrod's second tournament
                        actors[x].variables[y][13] = punishment;        // works as payoff from the last round
                    }
                    actors[x].defections[y]++;                          // works as S in Axelrod's second tournament
                    actors[x].variables[y][10] = 1;                     // works as J in Axelrod's second tournament
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y] == 3)
                    {
                        actors[x].defections[y] = 0;
                        actors[x].standing[y]++;                        // works as C in Axelrod's second tournament
                        actors[x].variables[y][10] = 0;
                    }
                    actors[x].variables[y][11] = 1;                     // works as JA in Axelrod's second tournament
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y] == 2)
                        actors[x].variables[y][11] = 0;
                    if (actors[x].variables[y][12] == 2)                    // works as FD in Axelrod's second tournament
                    {
                        actors[x].variables[y][12] = 0;
                        actors[x].array[y][1, 0] = (actors[x].array[y][1, 0] * actors[x].array[y][1, 1] + actors[x].variables[y][13]) / (actors[x].array[y][1, 1] + 1);
                        actors[x].array[y][1, 1]++;
                    }
                    if (actors[x].variables[y][12] == 1)                    // works as FD in Axelrod's second tournament
                    {
```

```
                        actors[x].variables[y][12] = 2;
                        actors[x].array[y][0, 0] = (actors[x].array[y][0, 0] * actors[x].
array[y][0, 1] + actors[x].variables[y][13]) / (actors[x].array[y][0, 1] + 1);
                        actors[x].array[y][0, 1]++;
                    }
                    if (Math.Abs(actors[x].variables[y][12] - 1.5) == 0.5)
                        return true;
                    if (actors[x].paramOne[y] < 0.75 * reward * actors[x].encounters[y])
                    {
                        if (actors[x].paramOne[y] < (7.0 / 12.0) * reward * actors[x].
encounters[y])
                        {
                            if (actors[x].variables[y][10] == 1)
                                return false;
                            return true;
                        }
                        if (rnd.NextDouble() <= 0.25 + actors[x].standing[y] / actors[x].
encounters[y] - actors[x].defections[y] * 0.25 + (actors[x].paramOne[y] - actors[x].
paramTwo[y]) / 100 + 4 / actors[x].encounters[y])
                            return true;
                        return false;
                    }
                    if (rnd.NextDouble() <= 0.95 - (actors[x].array[y][0, 0] + actors[x].
array[y][1, 0] - 5) / 15 + 1.0 / Math.Pow(actors[x].encounters[y], 2) - actors[x].
variables[y][10] / 4)
                        return true;
                    actors[x].variables[y][12] = 1;
                    return false;
                }
            }
            actors[x].array[y][0, 0] = 5;            // works as AD in Axelrod's second
tournament
            actors[x].array[y][0, 1] = 1;            // works as AK in Axelrod's second
tournament
            actors[x].array[y][1, 1] = 1;            // works as NK in Axelrod's second
tournament
            return true;
        }
        else if (strategyNo == 28) //GROFMAN
        {
            actors[x].encounters[y]++;
            int choice = 0;                 // works as MYOLD in Axelrod's code
            actors[x].standing[y] = 0;
            actors[x].defections[y] = 0;    // works as IPREV7 in Axelrod's code
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].encounters[y] < 8)
                    {
                        if (actors[x].encounters[y] < 3)
                            return true;
                        if (actors[x].interactions[a][y] == 1 || actors[x].interactions
[a][y] == 3)
                            return true;
                        return false;
                    }
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y]
 == 2)
                        choice = 1;
                    for (int b = a - 1; b >= 1; b--)
                    {
                        if (actors[x].interactions[b][y] != 0)
                        {
                            actors[x].standing[y]++;
                            if (actors[x].interactions[b][y] == 2 || actors[x].
interactions[b][y] == 4)
                                actors[x].defections[y]++;
                        }
```

```
                        if (actors[x].standing[y] == 7)
                            break;
                    }
                    if (choice == 1 && actors[x].defections[y] < 3)
                        return true;
                    if (choice == 1 && actors[x].defections[y] > 2)
                        return false;
                    if (choice == 0 && actors[x].defections[y] < 2)
                        return true;
                    if (choice == 0 && actors[x].defections[y] > 1)
                        return false;
                }
            }
            return true;
        }
        else if (strategyNo == 29) //JOSS
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[y]
 == 3)
                        actors[x].paramOne[y]++;                    // works as C in
Axelrod's second tournament
                    if (actors[x].standing[y] == 2)                 // works as S in
Axelrod's second tournament
                    {
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                        {
                            actors[x].defections[y]++;              // works as D in
Axelrod's second tournament
                            if (actors[x].defections[y] > 10)
                                actors[x].standing[y] = 3;
                        }
                        else
                            actors[x].defections[y] = 0;
                        return false;
                    }
                    else if (actors[x].standing[y] == 3)
                    {
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                        {
                            actors[x].defections[y]++;
                            if (actors[x].defections[y] > 20)
                            {
                                actors[x].defections[y] = 0;
                                return true;
                            }
                            return false;
                        }
                        else
                            actors[x].defections[y] = 0;
                        return true;
                    }
                    else if (actors[x].standing[y] == 4)
                    {
                        if (actors[x].interactions[a][y] == 1 || actors[x].interactions
[a][y] == 3)
                            actors[x].standing[y] = 1;
                        else
                        {
                            actors[x].paramTwo[y]++;                // works as F in
Axelrod's second tournament
                            if (actors[x].paramTwo[y] > 3)
                            {
```

```
                                actors[x].standing[y] = 3;
                                actors[x].defections[y] = 0;
                            }
                            else
                                actors[x].standing[y] = 1;
                        }
                        return true;
                    }
                    else if (actors[x].standing[y] == 1)
                    {
                        if (rnd.NextDouble() < 0.1)
                        {
                            actors[x].standing[y] = 5;
                            return false;
                        }
                    }
                    else if (actors[x].standing[y] == 5)
                        actors[x].standing[y] = 4;
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] ↙
 == 4)
                    {
                        actors[x].defections[y]++;
                        if (actors[x].defections[y] > 20)
                        {
                            actors[x].defections[y] = 0;
                            actors[x].standing[y] = 3;
                            return true;
                        }
                    }
                    else
                        actors[x].defections[y] = 0;
                    if (actors[x].paramOne[y] < 0.7 * (actors[x].encounters[y] - 3))
                    {
                        actors[x].standing[y] = 2;
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions ↙
[a][y] == 4)
                        {
                            actors[x].defections[y]++;
                            if (actors[x].defections[y] > 10)
                                actors[x].standing[y] = 3;
                        }
                        else
                            actors[x].defections[y] = 0;
                        return false;
                    }
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] ↙
 == 4)
                        return false;
                    return true;
                }
            }
            return true;
        }
        else if (strategyNo == 30) //PINKLEY
        {
            actors[x].encounters[y]++;
            double xx = 0.999;        // works as X in Axelrod's second tournament
            double px = 0.001;        // works as PX in Axelrod's second tournament
            double yy = 0.001;        // works as Y in Axelrod's second tournament
            double py = 0.999;        // works as PY in Axelrod's second tournament
            double z = 0.999;         // works as Z in Axelrod's second tournament
            double pz = 0.001;        // works as PZ in Axelrod's second tournament
            double w = 0.001;         // works as W in Axelrod's second tournament
            if (actors[x].encounters[y] == 1)
            {
                actors[x].array[y][0, 0] = 1.999;       // works as QC[1] in Axelrod's ↙
second tournament
                actors[x].array[y][0, 1] = 1.999;       // works as QC[2] in Axelrod's ↙
second tournament
```

```
                actors[x].array[y][1, 0] = 0.001;       // works as QC[3] in Axelrod's
second tournament
                actors[x].array[y][1, 1] = 0.001;       // works as QC[4] in Axelrod's
second tournament
                actors[x].defections[y] = 0;            // works as N in Axelrod's second
 tournament
                for (int a = 10; a < 14; a++)
                    actors[x].variables[y][a] = 2;      // works as QN[N] in Axelrod's
second tournament; a = N+10; variables [10] - [13]
            }
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].encounters[y] > 2)
                    {
                        if (actors[x].interactions[a][y] == 1 || actors[x].interactions
[a][y] == 3)
                            actors[x].array[y][(int)actors[x].defections[y] / 2, (int)
actors[x].defections[y] % 2]++;
                        actors[x].variables[y][(int)actors[x].defections[y] + 10]++;
                        xx = actors[x].array[y][0, 0] / actors[x].variables[y][10];
                        px = 1 - xx;
                        z = actors[x].array[y][0, 1] / actors[x].variables[y][11];
                        pz = 1 - z;
                        yy = actors[x].array[y][1, 0] / actors[x].variables[y][12];
                        py = 1 - yy;
                        w = actors[x].array[y][1, 1] / actors[x].variables[y][13];
                    }
                    if (actors[x].interactions[a][y] == 1)
                        actors[x].defections[y] = 0;
                    else if (actors[x].interactions[a][y] == 2)
                        actors[x].defections[y] = 1;
                    else if (actors[x].interactions[a][y] == 3)
                        actors[x].defections[y] = 2;
                    else
                        actors[x].defections[y] = 3;
                    break;
                }
            }
            double[] e = new double[11];
            for (int q = 0; q < 11; q++)
                e[q] = new double();                        // works as E(q - 1) in
Axelrod's second tournament
            e[0] = 3 * z / (z + px);
            e[1] = (3 * (yy * z + w * pz) + 5 * z * px + px * pz) / (yy * z + w * pz + px
 + z * px + px * pz);
            e[2] = (3 * yy * w + 5 * w * px + px * pz) / (yy * w + 2 * w * px + px * pz);
            e[3] = (3 * py * w + 5 * z * px + px * py) / (py * w + z * px + 2 * px * py);
            e[4] = (3 * z + 5 * xx * z + px * z) / (1 - xx * yy - w * px + 2 * z);
            e[5] = (8 * w * z + z * px) / (2 * w * z + w * py + z * px);
            e[6] = (4 * z * py + 5 * xx * z) / (2 * z * py + 0.999 * py + xx * z);
            e[7] = (3 * (yy * z + w * pz) + 5 * (z * 0.999 + w * xx) + 1 - xx * yy - z *
py) / (yy * z + w * pz + 2 - 2 * xx * yy - w * px + z * 0.999 + w * xx - z * py);
            e[8] = (3 * w * yy + 5 * w + 1 - xx * yy - z * py) / (2 * w + 1 - xx * yy - z
 * py);
            e[9] = (3 * w * py + 5 * (z * 0.999 + w * xx) + py) / (z * 0.999 + w * xx + 2
 * py);
            e[10] = (5 * w + py) / (w + py);
            int ibest = 0;                              // works as IBEST in Axelrod's
 second tournament
            double beste = e[0];                        // works as BESTE in Axelrod's
 second tournament
            for (int r = 1; r < 11; r++)
            {
                if (e[r] > beste)
                {
                    ibest = r;
                    beste = e[r];
```

```
                }
            }
            double[,] ipol = { { 0, 0, 0, 0 }, { 0, 1, 0, 0 }, { 0, 1, 0, 1 }, { 0, 1, 1, 0 }, { 1, 0, 0, 0 }, { 1, 0, 0, 1 }, { 1, 0, 1, 0 }, { 1, 1, 0, 0 }, { 1, 1, 0, 1 }, { 1, 1, 1, 0 }, { 1, 1, 1, 1 } };      // works as IPOL in Axelrod's second tournament
            if (ipol[ibest, (int)actors[x].defections[y]] == 0)
                return true;
            return false;
        }
        else if (strategyNo == 31) //NYDEGGER; as described in Ax´s article 1980: 22 is different than in Ax's code
        {
            actors[x].encounters[y]++;
            if (actors[x].encounters[y] == 1)
                actors[x].standing[y] = 0;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    actors[x].standing[y] = 4 * ((int)actors[x].standing[y] % 16);
                    if (actors[x].interactions[a][y] == 2)
                        actors[x].standing[y] += 1;
                    else if (actors[x].interactions[a][y] == 3)
                        actors[x].standing[y] += 2;
                    else if (actors[x].interactions[a][y] == 4)
                        actors[x].standing[y] += 3;
                    if (actors[x].encounters[y] < 4)
                    {
                        if (actors[x].encounters[y] == 3 && actors[x].standing[y] == 6)
                            return false;
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] == 4)
                            return false;
                        return true;
                    }
                    else
                    {
                        if (actors[x].standing[y] == 0 || actors[x].standing[y] == 27 || actors[x].standing[y] == 28 || actors[x].standing[y] == 32 || actors[x].standing[y] == 40 || actors[x].standing[y] == 41 || actors[x].standing[y] == 42 || actors[x].standing[y] == 43 || actors[x].standing[y] == 44 || actors[x].standing[y] == 46 || actors[x].standing[y] == 47 || actors[x].standing[y] == 48 || actors[x].standing[y] == 56 || actors[x].standing[y] == 57 || actors[x].standing[y] == 59 || actors[x].standing[y] == 60 || actors[x].standing[y] == 62 || actors[x].standing[y] == 63)
                            return true;
                        return false;
                    }
                }
            }
            return true;
        }
        else if (strategyNo == 32) //PEBLEY
        {
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] == 1)
                    return true;
                else if (actors[x].interactions[a][y] == 4)
                    return false;
                else if (actors[x].interactions[a][y] != 0)
                {
                    if (rnd.NextDouble() > 0.2)
                        return false;
                    return true;
                }
            }
            return true;
        }
        else if (strategyNo == 33) //FALK & LANGSTED
```

```
            {
                actors[x].encounters[y]++;
                int k85r = new int();
                int J = new int();
                for (int q = actors[x].interactions.Count - 1; q >= 1; q--)
                {
                    if (actors[x].interactions[q][y] != 0)
                    {
                        k85r = 1;
                        if (actors[x].interactions[q][y] == 2 || actors[x].interactions[q][y]
 == 1)
                            k85r = 0;
                        J = 0;
                        if (actors[x].interactions[q][y] == 2 || actors[x].interactions[q][y]
 == 4)
                            J = 1;
                        actors[x].paramOne[y] = (actors[x].paramOne[y] % 1e7) * 10 + J;
                        actors[x].paramTwo[y] = (actors[x].paramTwo[y] % 1e7) * 10 + k85r;
                        if (actors[x].variables[y][11] == 0)        // works as F1 in Axelrod
's second tournament
                        {
                            if (J == 0)
                                actors[x].array[y][1, 0]++;         // works as I3 in Axelrod
's second tournament
                            else
                                actors[x].array[y][1, 1]++;         // works as I4 in Axelrod
's second tournament
                        }
                        else
                        {
                            if (J == 0)
                                actors[x].array[y][0, 0]++;         // works as I1 in Axelrod
's second tournament
                            else
                                actors[x].array[y][0, 1]++;         // works as I2 in Axelrod
's second tournament
                        }
                        if (actors[x].encounters[y] > 20)
                        {
                            double a = (actors[x].array[y][0, 0] + 1e-6) / (actors[x].array
[y][0, 1] + 1e-6);
                            double b = (actors[x].array[y][1, 0] + 1e-6) / (actors[x].array
[y][1, 1] + 1e-6);
                            if (a <= 1.5 && a >= 0.5 && b <= 1.5 && b >= 0.5)
                            {
                                actors[x].variables[y][11] = k85r;
                                actors[x].defections[y] = 0;        // works as D in Axelrod
's second tournament
                                return false;
                            }
                        }
                        if (actors[x].variables[y][10] == 1)        // works as T in Ax's
code
                        {
                            if (J == 0)
                                return true;
                            return false;
                        }
                        if (actors[x].paramOne[y] == 11111111)
                        {
                            actors[x].variables[y][10] = 1;
                            if (J == 0)
                                return true;
                            return false;
                        }
                        if (actors[x].standing[y] == 1)         // works as C in Axelrod's
second tournament
                        {
                            actors[x].standing[y] = 0;
```

```
                        actors[x].variables[y][11] = k85r;
                        return true;
                    }
                    if (actors[x].encounters[y] <= 30 && (actors[x].paramOne[y] % 100) ==↙
 11)
                        break;
                    if (actors[x].paramOne[y] % 10000 == 1011 && actors[x].paramTwo[y] % ↙
10000 == 111)
                    {
                        actors[x].standing[y] = 1;
                        actors[x].variables[y][11] = k85r;
                        return true;
                    }
                    if (actors[x].array[y][0, 1] + actors[x].array[y][1, 1] >= actors[x].↙
array[y][0, 0] + actors[x].array[y][0, 1] + 3)
                    {
                        actors[x].variables[y][11] = k85r;
                        actors[x].defections[y] = 0;
                        return false;
                    }
                    if (k85r == 1 && J == 0)
                    {
                        actors[x].variables[y][11] = k85r;
                        actors[x].defections[y] = 0;
                        return false;
                    }
                    if (actors[x].defections[y] == 1)
                    {
                        actors[x].variables[y][11] = k85r;
                        actors[x].defections[y] = 0;
                        return true;
                    }
                    if (k85r == 0 && J == 0)
                    {
                        actors[x].variables[y][11] = k85r;
                        return true;
                    }
                    break;
                }
            }
            if (actors[x].encounters[y] == 1)
            {
                actors[x].standing[y] = 0;
                return true;
            }
            if (k85r == 0 && J == 1)
            {
                actors[x].variables[y][11] = k85r;
                actors[x].defections[y] = 0;
                return false;
            }
            if (k85r == 1)
            {
                actors[x].variables[y][11] = k85r;
                actors[x].defections[y] = 1;
                return false;
            }
            actors[x].variables[y][11] = k85r;
            return true;
        }
        else if (strategyNo == 34) //WEIDERMAN
        {
            if (actors[x].standing[y] == 0)
                return false;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].encounters[y] >= 4)       //"encounters" functions as  ↙
```

```
JHIS in the Axelrod's second tournament.
                        actors[x].encounters[y] -= 4;
                    actors[x].encounters[y] *= 2;
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] ↙
 == 4)
                        actors[x].encounters[y] += 1;
                    if (actors[x].encounters[y] == 7)
                    {
                        actors[x].standing[y] = 0;
                        return false;
                    }
                    return true;
                }
            }
            return true;
        }
        else if (strategyNo == 35) //ADAMS ROBERT
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    actors[x].standing[y]++;
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] ↙
 == 4)
                    {
                        actors[x].paramTwo[y]++;    // works as W in Axelrod's second    ↙
tournament
                        actors[x].paramOne[y] = actors[x].paramOne[y] / 2;
                    }
                    if (actors[x].encounters[y] == 2)
                        return true;
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] ↙
 == 4)
                    {
                        actors[x].paramTwo[y]++;
                        if ((actors[x].paramTwo[y] > 2 && actors[x].paramTwo[y] % 3 == 0) ↙
 || (actors[x].paramTwo[y] - 1) % 3 == 0)
                        {
                            actors[x].standing[y] = 1;
                            actors[x].paramOne[y] = actors[x].paramOne[y] / 2;
                            return false;
                        }
                        if (rnd.NextDouble() >= actors[x].paramOne[y])
                        {
                            actors[x].paramOne[y] = actors[x].paramOne[y] / 2;
                            return false;
                        }
                        actors[x].paramOne[y] = actors[x].paramOne[y] / 2;
                        return true;
                    }
                    if (actors[x].standing[y] == 1 || actors[x].standing[y] == 2)
                        return false;
                    if (actors[x].paramTwo[y] > 2 && (actors[x].paramTwo[y] % 3 == 0 ||   ↙
(actors[x].paramTwo[y] - 1) % 3 == 0))
                    {
                        actors[x].standing[y] = 1;
                        actors[x].paramOne[y] = actors[x].paramOne[y] / 2;
                        return false;
                    }
                    return true;
                }
            }
            actors[x].standing[y] = 4;          // works as S in Axelrod's second        ↙
tournament
            actors[x].paramOne[y] = 0.8;        // works as Q in Axelrod's second        ↙
tournament
            return true;
```

```
            }
            else if (strategyNo == 36) //DAWES & BATELL
            {
                if (actors[x].standing[y] == 0)
                    return false;
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                        {
                            actors[x].defections[y]++;
                            actors[x].standing[y] = actors[x].encounters[y] - actors[x].
defections[y];
                            if (Math.Pow(1.6667, actors[x].defections[y]) * Math.Pow(0.882,
actors[x].standing[y]) >= 5)
                            {
                                actors[x].standing[y] = 0;
                                return false;
                            }
                        }
                        return true;
                    }
                }
                return true;
            }
            else if (strategyNo == 37) //LEFEVRE
            {
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                            actors[x].defections[y]++;
                        break;
                    }
                }
                if (5 * actors[x].defections[y] > actors[x].encounters[y])
                    return false;
                return true;
            }
            else if (strategyNo == 38) //ANDERSON
            {
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].encounters[y] != 2)
                        {
                            if (actors[x].paramTwo[y] == 0 && (actors[x].interactions[a][y] =
= 1 || actors[x].interactions[a][y] == 3))
                                actors[x].array[y][0, 0]++;                 // works as NCC
in Axelrod's second tournament
                            else if (actors[x].paramTwo[y] == 0 && (actors[x].interactions[a]
[y] == 2 || actors[x].interactions[a][y] == 4))
                                actors[x].array[y][0, 1]++;                 // works as NCD
in Axelrod's second tournament
                            else if (actors[x].paramTwo[y] == 1 && (actors[x].interactions[a]
[y] == 1 || actors[x].interactions[a][y] == 3))
                                actors[x].array[y][1, 0]++;                 // works as NDC
in Axelrod's second tournament
                            else if (actors[x].paramTwo[y] == 1 && (actors[x].interactions[a]
[y] == 2 || actors[x].interactions[a][y] == 4))
                                actors[x].array[y][1, 1]++;                 // works as NDD
```

```
in Axelrod's second tournament
                    }
                    actors[x].paramTwo[y] = 1;
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y]
 == 2)
                        actors[x].paramTwo[y] = 0;                      // works as IOLD2
 in Axelrod's second tournament
                    if (actors[x].encounters[y] < 16)
                    {
                        if ((actors[x].interactions[a][y] == 1 || actors[x].interactions
[a][y] == 3) || actors[x].defections[y] >= 3)
                            return true;
                        actors[x].defections[y]++;
                        return false;
                    }
                    if (actors[x].encounters[y] == 17 && (actors[x].interactions[a][y] ==
 2 || actors[x].interactions[a][y] == 4) && actors[x].array[y][0, 1] == 1 && actors[x].
array[y][1, 1] == 0)
                        actors[x].standing[y] = 0;
                    if (actors[x].array[y][0, 1] * 3 >= actors[x].array[y][0, 0] + actors
[x].array[y][0, 1])
                        return false;
                    if (actors[x].encounters[y] % 4 != 0 || actors[x].standing[y] == 0)
                        return true;
                    if (actors[x].array[y][1, 0] >= (int)(actors[x].encounters[y] / 12) |
| actors[x].array[y][1, 1] == 0)
                        return false;
                    return true;
                }
            }
            return true;
        }
        else if (strategyNo == 39) //DOWNING
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].encounters[y] > 2)
                    {
                        if (actors[x].standing[y] == 0)
                        {
                            if (actors[x].interactions[a][y] == 1 || actors[x].
interactions[a][y] == 3)
                                actors[x].paramTwo[y]++;
                            actors[x].array[y][1, 1]++;
                            actors[x].array[y][1, 0] = actors[x].paramTwo[y] / actors[x].
array[y][1, 1];
                        }
                        else
                        {
                            if (actors[x].interactions[a][y] == 1 || actors[x].
interactions[a][y] == 3)
                                actors[x].paramOne[y]++;
                            actors[x].array[y][0, 0]++;
                            actors[x].array[y][0, 1] = actors[x].paramOne[y] / actors[x].
array[y][0, 0];
                        }
                    }
                    actors[x].standing[y] = 0;
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y]
 == 2)
                        actors[x].standing[y] = 1;
                    double c = 6 * actors[x].array[y][0, 1] - 8 * actors[x].array[y][1,
0] - 2;
                    double alt = 4 * actors[x].array[y][0, 1] - 5 * actors[x].array[y][1,
 0] - 1;
                    if (c >= 0 && c >= alt)
```

```
                        return true;
                    if (c >= 0 && c < alt)
                    {
                        if (actors[x].standing[y] == 1)
                            return false;
                        return true;
                    }
                    if (alt >= 0)
                    {
                        if (actors[x].standing[y] == 1)
                            return false;
                        return true;
                    }
                    return false;
                }
            }
            actors[x].array[y][0, 1] = 1;
            return true;
        }
        else if (strategyNo == 40) //ZIMMERMAN
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].interactions[a][y] == 1)
                    {
                        if (actors[x].rewardshift[y] - 1 < 0)
                            actors[x].defections[y] += (reward - (reward - punishment) * ↙
(1 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));     // actors[x].  ↙
defections[y] works as K in Axelrod's code (score of player x gained from interactions   ↙
with actor y so far)
                        else if (actors[x].rewardshift[y] - 1 >= 0)
                            actors[x].defections[y] += (reward + (temptation - reward) * ↙
(1 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                    }
                    else if (actors[x].interactions[a][y] == 2)
                        actors[x].defections[y] += sucker;
                    else if (actors[x].interactions[a][y] == 3)
                        actors[x].defections[y] += temptation;
                    else
                        actors[x].defections[y] += punishment;          // works as K in ↙
Axelrod's second tournament
                    if (actors[x].standing[y] == 0 && (actors[x].interactions[a][y] == 2 ↙
|| actors[x].interactions[a][y] == 4))
                        return false;
                    if (actors[x].standing[y] == 1 && (actors[x].interactions[a][y] == 1 ↙
|| actors[x].interactions[a][y] == 3))
                        return true;
                    if (actors[x].standing[y] == 1)         // works as ITEST in Axelrod ↙
's second tournament
                        actors[x].array[y][0, 1]++;         // works as IDUNU in Axelrod ↙
's second tournament
                    else
                        actors[x].array[y][0, 0]++;         // works as IDUNB in Axelrod ↙
's second tournament
                    if (actors[x].array[y][0, 1] < actors[x].paramTwo[y] && actors[x].   ↙
array[y][0, 0] < actors[x].paramOne[y])
                    {
                        if (actors[x].standing[y] == 1)
                            return true;
                        return false;
                    }
                    actors[x].array[y][0, 1] = 0;
                    actors[x].array[y][0, 0] = 0;
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]↙
 == 4)
                        actors[x].standing[y] = 0;
```

```
                    else
                        actors[x].standing[y] = 1;
                    if (actors[x].standing[y] == 0)
                    {
                        actors[x].paramOne[y] = (int)(1.6667 * (actors[x].paramTwo[y] +
1));
                        return false;
                    }
                    actors[x].paramTwo[y] = actors[x].paramTwo[y] - reward + (int)(actors
[x].defections[y] / actors[x].encounters[y]);
                    if (actors[x].paramTwo[y] <= 0)
                        actors[x].paramTwo[y] = 1;
                    return true;
                }
            }
            actors[x].paramOne[y] = 8;      // works as IPAYB in Axelrod's tournament
            actors[x].paramTwo[y] = 4;      // works as IAGGD in Axelrod's tournament
            return true;
        }
        else if (strategyNo == 41) //NEWMAN
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].encounters[y] > 2)
                    {
                        if (actors[x].paramTwo[y] == 0)                 // works as IOLD
in Axelrod's second tournament
                        {
                            if (actors[x].interactions[a][y] == 1 || actors[x].
interactions[a][y] == 3)
                                actors[x].array[y][0, 0]++;             // works as QCA
in Axelrod's second tournament
                            actors[x].array[y][0, 1]++;                 // works as QNA
in Axelrod's second tournament
                            actors[x].standing[y] = actors[x].array[y][0, 0] / actors[x].
array[y][0, 1];    // works as ALPHA in Axelrod's second tournament
                        }
                        else
                        {
                            if (actors[x].interactions[a][y] == 1 || actors[x].
interactions[a][y] == 3)
                                actors[x].array[y][1, 0]++;             // works as QCB
in Axelrod's second tournament
                            actors[x].array[y][1, 1]++;                 // works as QNB
in Axelrod's second tournament
                            actors[x].paramOne[y] = actors[x].array[y][1, 0] / actors[x].
array[y][1, 1];    // works as BETA in Axelrod's second tournament
                        }
                    }
                    actors[x].paramTwo[y] = 1;
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y]
 == 2)
                        actors[x].paramTwo[y] = 0;
                    if (6 * actors[x].standing[y] - 9 * actors[x].paramOne[y] - 2 >= 0)
                    {
                        if (6 * actors[x].standing[y] - 9 * actors[x].paramOne[y] - 2 >=
4 * actors[x].standing[y] - 6 * actors[x].paramOne[y] - 1)
                            return true;
                        if (1 - actors[x].paramTwo[y] == 0)
                            return true;
                        return false;
                    }
                    if (4 * actors[x].standing[y] - 6 * actors[x].paramOne[y] - 1 >= 0)
                    {
                        if (1 - actors[x].paramTwo[y] == 0)
                            return true;
```

```
                        return false;
                    }
                    if (actors[x].paramTwo[y] == 0 || (actors[x].interactions[a][y] == 1 ↙
|| actors[x].interactions[a][y] == 3))
                    {
                        actors[x].defections[y] = 0;                // works as MUTDEF in↙
 Axelrod's second tournament
                        return false;
                    }
                    actors[x].defections[y]++;
                    if (actors[x].defections[y] > 3)
                        return true;
                    return false;
                }
            }
            return true;
        }
        else if (strategyNo == 42) //JONES
        {
            actors[x].encounters[y]++;
            int k81r = new int();
            int J = new int();
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    J = 0;
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]↙
 == 4)
                        J = 1;
                    if (actors[x].interactions[a][y] == 1)
                    {
                        if (actors[x].rewardshift[y] - 1 < 0)
                        {
                            actors[x].paramOne[y] += (reward - (reward - punishment) * (1↙
 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));
                            actors[x].paramTwo[y] += (reward - (reward - punishment) * (1↙
 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));
                        }
                        else if (actors[x].rewardshift[y] - 1 >= 0)
                        {
                            actors[x].paramOne[y] += (reward + (temptation - reward) * (1↙
 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                            actors[x].paramTwo[y] += (reward + (temptation - reward) * (1↙
 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                        }
                    }
                    else if (actors[x].interactions[a][y] == 2)
                    {
                        actors[x].paramOne[y] += sucker;
                        actors[x].paramTwo[y] += temptation;
                    }
                    else if (actors[x].interactions[a][y] == 3)
                    {
                        actors[x].paramOne[y] += temptation;
                        actors[x].paramTwo[y] += sucker;
                    }
                    else
                    {
                        actors[x].paramOne[y] += punishment;                        //    ↙
works as K in Axelrod's second tournament (IS)
                        actors[x].paramTwo[y] += punishment;                        //    ↙
works as L in Axelrod's second tournament (JS)
                    }
                    if (actors[x].encounters[y] == 81 && actors[x].paramOne[y] == actors ↙
[x].paramTwo[y] && actors[x].rewardshift[y] == 74)
                        actors[x].variables[y][10] = 1;                             //    ↙
works as T0 in Axelrod's second tournament
                    if (actors[x].encounters[y] == 2 && J == 1)
```

```
                        actors[x].defections[y] = 9;
                    if (actors[x].encounters[y] < actors[x].defections[y])
                    {
                        k81r = J;
                        break;
                    }
                    else
                    {
                        if (actors[x].variables[y][12] > 7)
                            actors[x].variables[y][12] -= 8;
                        if (J == 0)
                            actors[x].variables[y][14 + (int)actors[x].variables[y][12]]+↙
+;
                        if (actors[x].defections[y] == 9 && actors[x].variables[y][10] ==↙
 1)
                        {
                            if (J == 1)
                                actors[x].variables[y][10] = 0;
                            else
                            {
                                int T2 = 0;
                                while (((actors[x].encounters[y] <= 80 + T2) || (actors ↙
[x].encounters[y] >= 140 + T2)) && ((actors[x].encounters[y] < 140 + T2) || (actors[x]. ↙
encounters[y] > 180 + T2)))
                                    T2 += 100;
                                if ((actors[x].encounters[y] <= 80 + T2) || (actors[x]. ↙
encounters[y] >= 140 + T2))
                                    k81r = 0;
                                else
                                    k81r = 1;
                                break;
                            }
                        }
                        if (J == 0)
                        {
                            if (actors[x].variables[y][13] < 0 || actors[x].variables[y] ↙
[13] >= 6)      // works as T8 in Axelrod's second tournament
                            {
                                if (actors[x].variables[y][13] > 0)
                                    actors[x].variables[y][13] = -200;
                                actors[x].variables[y][13]++;
                                k81r = 0;
                                break;
                            }
                            actors[x].variables[y][13] = 0;
                        }
                        else
                        {
                            actors[x].variables[y][13]++;
                            if (actors[x].variables[y][13] < 8 || actors[x].variables[y] ↙
[13] > 9)
                            {
                                if (actors[x].variables[y][13] > 1)
                                    actors[x].variables[y][13] = 1;
                            }
                            else
                            {
                                k81r = 0;
                                break;
                            }
                        }
                        if (actors[x].paramTwo[y] > actors[x].paramOne[y] + actors[x]. ↙
standing[y])
                        {
                            k81r = J;
                            break;
                        }
                        actors[x].array[y][1, 1] = actors[x].variables[y][11];
                        if (actors[x].array[y][1, 1] > 7)
```

```
                                actors[x].array[y][1, 1] -= 8;                          //
works as D4 in Axelrod's second tournament
                            double a1 = actors[x].variables[y][14 + (int)actors[x].array[y][1
, 1]];
                            double a2 = actors[x].variables[y][22 + (int)actors[x].array[y][1
, 1]];
                            if (a2 == 0)
                                a2 = 1;
                            actors[x].array[y][0, 0] = 3 * a1 / a2;                     //
works as A in Axelrod's second tournament
                            actors[x].array[y][0, 1] = 4 * a1 / a2 + 1;                 //
works as B in Axelrod's second tournament
                            for (int c = 0; c < 4; c++)
                            {
                                actors[x].variables[y][30 + c] = actors[x].array[y][0, 0];
                                actors[x].variables[y][34 + c] = actors[x].array[y][0, 1];
                            }
                            int e0 = 5;
                            int e1 = 6;
                            int f0 = 3;
                            int f2 = 7;
                            for (int h = 0; h < 2; h++)
                            {
                                if (actors[x].variables[y][11] > 4)
                                    actors[x].variables[y][11] -= 4;
                                actors[x].variables[y][11] *= 2;
                                for (int c = 0; c < 8; c++)
                                {
                                    actors[x].array[y][1, 1] = actors[x].variables[y][11];
                                    if (c == e0 || c == e1 || c == 7 || c == 8)
                                        actors[x].array[y][1, 1] = actors[x].variables[y][11]
 + 1;
                                    if (actors[x].array[y][1, 1] == 9)
                                        actors[x].array[y][1, 1] = 1;
                                    if (actors[x].array[y][1, 1] > 7)
                                        actors[x].array[y][1, 1] -= 8;
                                    a1 = actors[x].variables[y][14 + (int)actors[x].array[y]
[1, 1]];
                                    a2 = actors[x].variables[y][22 + (int)actors[x].array[y]
[1, 1]];
                                    if (a2 == 0)
                                        a2 = 1;
                                    actors[x].array[y][0, 0] = 3 * a1 / a2;
                                    actors[x].array[y][0, 1] = 4 * a1 / a2 + 1;
                                    if (c == f0 || c == 4 || c == f2 || c == 8)
                                        actors[x].variables[y][30 + c] += actors[x].array[y]
[0, 1];
                                    else
                                        actors[x].variables[y][30 + c] += actors[x].array[y]
[0, 0];
                                }
                                if (h == 0)
                                {
                                    e0 = 3;
                                    e1 = 4;
                                    f0 = 2;
                                    f2 = 6;
                                }
                            }
                            double s = 0;
                            for (int z = 0; z < 8; z++)
                            {
                                if (actors[x].variables[y][30 + z] > s)
                                {
                                    s = actors[x].variables[y][30 + z];
                                    actors[x].array[y][1, 0] = z + 1;                   //
works as S1 in Axelrod's second tournament
                                }
                            }
```

```
                        k81r = 1;
                        if (actors[x].array[y][1, 0] < 5)
                            k81r = 0;
                        break;
                    }
                }
            }
            if (actors[x].encounters[y] == 1)
            {
                actors[x].standing[y] = 25;                                         //
works as T6 in Axelrod's second tournament
                actors[x].defections[y] = 5;                                        //
works as T9 in Axelrod's second tournament
                k81r = 0;
            }
            actors[x].variables[y][12] = actors[x].variables[y][11];                //
works as T5 in Axelrod's second tournament
            if (actors[x].encounters[y] % 10 == 0)
            {
                for (int c = 0; c < 8; c++)
                    actors[x].variables[y][14 + c] = actors[x].variables[y][14 + c] * 9;
   // works as L4() in Axelrod's second tournament; VARIABLES[14] - [21]
                actors[x].standing[y]++;
            }
            if (actors[x].variables[y][11] > 7)
                actors[x].variables[y][11] -= 8;
            if (actors[x].encounters[y] > 3)
                actors[x].variables[y][22 + (int)actors[x].variables[y][11]]++;
            if (actors[x].variables[y][11] > 4)
                actors[x].variables[y][11] -= 4;
            actors[x].variables[y][11] = actors[x].variables[y][11] * 2 + k81r;     //
works as T4 in Axelrod's second tournament
            if (k81r == 0)
                return true;
            return false;
        }
        else if (strategyNo == 43) //SHURMANN
        {
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    actors[x].paramOne[y] = 0;
                    if (actors[x].interactions[a][y] == 3 || actors[x].interactions[a][y]
 == 4)
                        actors[x].paramOne[y] = 1;
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                        actors[x].encounters[y] = actors[x].paramOne[y] + 3;    // works
as S and H in Axelrod's second tournament, respectively
                    else
                    {
                        actors[x].encounters[y] = actors[x].paramOne[y] + 1;
                        if (actors[x].paramTwo[y] == 0)                         // works
as Z in Axelrod's second tournament
                            return true;
                    }
                    actors[x].paramTwo[y] = 1;
                    if (actors[x].encounters[y] <= 1)
                        actors[x].defections[y] = actors[x].defections[y] * 0.57 + 0.43;
   // works as Q6 in Axelrod's second tournament
                    else if (actors[x].encounters[y] != 4)
                        actors[x].defections[y] = 0.5 * actors[x].defections[y];
                    else
                        actors[x].defections[y] = 0.74 * actors[x].defections[y] + 0.104;
                    if (rnd.NextDouble() > actors[x].defections[y])
                        return false;
                    return true;
                }
```

```
                }
                actors[x].defections[y] = 0.5;      // works as Q6 in Axelrod's second
tournament
                return true;
            }
            else if (strategyNo == 44) //NUSSBACHER
            {
                actors[x].encounters[y]++;
                if (actors[x].encounters[y] < 11)
                    return true;
                actors[x].defections[y] = 0;
                actors[x].paramOne[y] = 0;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        actors[x].paramOne[y]++;
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                            actors[x].defections[y]++;
                        if (actors[x].paramOne[y] >= 10)
                        {
                            if (actors[x].defections[y] > 8.9)
                            {
                                if (rnd.NextDouble() < 0.94)
                                    return false;
                                return true;
                            }
                            else if (actors[x].defections[y] == 7 || actors[x].defections[y]
== 6 || actors[x].defections[y] == 5 || actors[x].defections[y] == 2)
                            {
                                if (rnd.NextDouble() < 0.87)
                                    return false;
                                return true;
                            }
                            else if (actors[x].defections[y] == 4 || actors[x].defections[y]
== 3 || actors[x].defections[y] == 8)
                            {
                                if (rnd.NextDouble() < 0.915)
                                    return false;
                                return true;
                            }
                            else if (actors[x].defections[y] == 1)
                            {
                                if (rnd.NextDouble() < 0.23)
                                    return false;
                                return true;
                            }
                            return true;
                        }
                    }
                }
                return true;
            }
            else if (strategyNo == 45) //GLADSTEIN
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].standing[y] == 0)
                        {
                            if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                                return false;
                            return true;
                        }
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
```

```
                        {
                            actors[x].standing[y] = 0;
                            return true;
                        }
                        actors[x].standing[y]++;                        // stands for DC in ↙
Ax's tournament
                        if (actors[x].defections[y] == 1)
                            actors[x].encounters[y]++;                  // stands for MDC in ↙
Ax's tournament
                        actors[x].defections[y] = 1;
                        if (actors[x].encounters[y] / actors[x].standing[y] >= 0.5)
                            actors[x].defections[y] = 0;
                        if (actors[x].defections[y] == 1)
                            return true;
                        return false;
                    }
                }
                return false;
            }
            else if (strategyNo == 46) //BATELL
            {
                if (actors[x].defections[y] >= 10)
                    return false;
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] == ↙
4)
                    {
                        actors[x].defections[y]++;
                        if (actors[x].defections[y] >= 10)
                            return false;
                        if (actors[x].paramOne[y] != 0)
                        {
                            if (actors[x].encounters[y] - actors[x].paramOne[y] <= 4)
                            {
                                actors[x].defections[y] = 10;
                                return false;
                            }
                        }
                        actors[x].paramOne[y] = actors[x].encounters[y];
                        return true;
                    }
                    else if (actors[x].interactions[a][y] != 0)
                        return true;
                }
                return true;
            }
            else if (strategyNo == 47) //SMITH D. A.
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        actors[x].defections[y]++;      // number of defections in a row.     ↙
Works as D9 in Axelrod's code
                        if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y]↙
 == 3)
                            actors[x].defections[y] = 0;
                        break;
                    }
                }
                if (actors[x].defections[y] > 1)
                {
                    if (actors[x].defections[y] >= (5 + 3 * actors[x].encounters[y]))
                    {
                        actors[x].defections[y] = 0;
                        actors[x].encounters[y]++;      // works as D8 in Axelrod's code
                    }
```

```
                    if (rnd.NextDouble() <= 0.05)
                        return true;
                    return false;
                }
                if (rnd.NextDouble() <= 0.05)
                    return false;
                return true;
            }
            else if (strategyNo == 48) //LEYLAND
            {
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        actors[x].paramOne[y] = 0;
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                        {
                            actors[x].paramOne[y] = 1;          // works as J in Axelrod's
second tournament
                            actors[x].array[y][1, 1]++;         // works as I5 in Axelrod's
second tournament
                            actors[x].array[y][1, 0]++;         // works as D4 in Axelrod's
second tournament
                        }
                        if (actors[x].paramOne[y] == 0 && actors[x].array[y][1, 1] > 1)
                        {
                            if (actors[x].array[y][1, 1] > 5)
                            {
                                actors[x].array[y][1, 1] = 0;
                                actors[x].array[y][0, 0] = 0;   // works as I1 in Axelrod's
second tournament
                                return true;
                            }
                            actors[x].array[y][1, 1] = 0;
                        }
                        if (actors[x].encounters[y] < 30)
                        {
                            if (actors[x].paramOne[y] == 1)
                                return false;
                            return true;
                        }
                        if (actors[x].standing[y] == 0)         // works as I3 in Axelrod's
second tournament
                        {
                            actors[x].array[y][0, 1] = 0;       // works as I2 in Axelrod's
second tournament
                            if (actors[x].array[y][1, 1] <= 5)
                                actors[x].standing[y] = 1;
                            actors[x].array[y][1, 1] = 0;
                            actors[x].array[y][0, 0] = 0;
                            return true;
                        }
                        if (Math.Abs(actors[x].array[y][1, 0] / (actors[x].encounters[y] - 1)
 - 0.5) < 0.1)
                            actors[x].defections[y] -= 0.2;
                        if (actors[x].array[y][0, 1] == 1)
                        {
                            actors[x].array[y][0, 1] = 0;
                            if (actors[x].paramOne[y] == 1)
                            {
                                actors[x].defections[y] += 0.15;
                                if (actors[x].defections[y] > 1)
                                    actors[x].defections[y] = 1;
                                if (actors[x].array[y][1, 1] > 5)
                                    actors[x].standing[y] = 0;
                                actors[x].array[y][1, 1] = 0;
                                actors[x].array[y][0, 0] = 0;
```

```
                                return true;
                            }
                            actors[x].defections[y] -= 0.05;
                            if (actors[x].defections[y] < 0)
                                actors[x].defections[y] = 0;
                            if (actors[x].defections[y] >= 0.3)
                            {
                                if (actors[x].paramOne[y] == 1)
                                    return false;
                                return true;
                            }
                        }
                        if (rnd.NextDouble() > actors[x].defections[y])
                        {
                            actors[x].array[y][0, 0] = 1;
                            return false;
                        }
                        actors[x].array[y][0, 1] = actors[x].array[y][0, 0];
                        if (actors[x].paramOne[y] == 1)
                            return false;
                        return true;
                    }
                }
                actors[x].defections[y] = 0.75;              // works as X in Axelrod's
second tournament
                return true;
            }
            else if (strategyNo == 49) //MCGURRIN
            {
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].encounters[y] > 3)
                        {
                            if (actors[x].array[y][1, 0] == 1)
                            {
                                if (actors[x].interactions[a][y] == 2 || actors[x].
interactions[a][y] == 4)
                                    return false;
                                return true;
                            }
                            if (actors[x].array[y][0, 1] == 1)         // works as B in
Axelrod's tournament
                            {
                                if (actors[x].paramOne[y] == 1 && (actors[x].interactions[a]
[y] == 2 || actors[x].interactions[a][y] == 4))
                                    return false;
                                actors[x].paramOne[y] = 0;                  // works as JOLD
in Axelrod's second tournament
                                if (actors[x].interactions[a][y] == 2 || actors[x].
interactions[a][y] == 4)
                                    actors[x].paramOne[y] = 1;
                                return true;
                            }
                            if (actors[x].array[y][0, 0] == 1)
                            {
                                actors[x].paramTwo[y]++;                  // works as E in
Axelrod's second tournament
                                if (actors[x].paramTwo[y] != 8)
                                {
                                    if (actors[x].paramOne[y] != 1 || (actors[x].interactions
[a][y] == 1 || actors[x].interactions[a][y] == 3))
                                    {
                                        actors[x].paramOne[y] = 0;
                                        if (actors[x].interactions[a][y] == 2 || actors[x].
interactions[a][y] == 4)
                                            actors[x].paramOne[y] = 1;
```

```
                                    return true;
                                }
                            }
                            else
                                actors[x].paramTwo[y] = 0;
                            actors[x].paramOne[y] = 0;
                            if (actors[x].interactions[a][y] == 2 || actors[x].
interactions[a][y] == 4)
                                actors[x].paramOne[y] = 1;
                            return false;
                        }
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                        {
                            actors[x].array[y][1, 0] = 1;
                            if (actors[x].array[y][1, 1] == 1)
                                return false;
                        }
                        else
                            actors[x].array[y][0, 1] = 1;
                        return true;
                    }
                    if (actors[x].encounters[y] == 2)
                    {
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                            actors[x].array[y][1, 1] = 1;               // works as D in
Axelrod's tournament
                    }
                    else
                    {
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                        {
                            if (actors[x].array[y][1, 1] == 1)
                                actors[x].array[y][1, 0] = 1;           // works as C in
Axelrod's tournament
                        }
                        else
                        {
                            if (actors[x].array[y][1, 1] != 1)
                                actors[x].array[y][0, 0] = 1;           // works as A in
Axelrod's tournament
                        }
                    }
                    return true;
                }
            }
            return false;
        }
        else if (strategyNo == 50) //HOLLANDER
        {
            actors[x].encounters[y]++;
            if (actors[x].encounters[y] == 1)
            {
                actors[x].defections[y] = 1;        //defection works here as an index of
 the penultimae move of the oponent
                actors[x].standing[y] = (int)(23 * rnd.NextDouble() + 1);      //
standing is used to function as a IRAN in Axelrod's second tournament
            }
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]
 == 4)
                    {
                        if (actors[x].defections[y] == 0 && actors[x].encounters[y] !=
actors[x].standing[y])
```

```
                            return false;
                        actors[x].defections[y] = 0;
                    }
                    else
                        actors[x].defections[y] = 1;
                    break;
                }
            }
            if (actors[x].encounters[y] == actors[x].standing[y])
            {
                actors[x].standing[y] = (int)(23 * rnd.NextDouble() + actors[x].
encounters[y] + 1);
                return false;
            }
            return true;
        }
        else if (strategyNo == 51) //GRIM or FRIEDMAN
        {
            if (actors[x].standing[y] == 0)
                return false;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] ==
4)
                {
                    actors[x].standing[y] = 0;
                    return false;
                }
                else if (actors[x].interactions[a][y] != 0)
                    return true;
            }
            return true;
        }
        else if (strategyNo == 52) //HUFFORD GEORGE
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].interactions[a][y] == 1)
                    {
                        if (actors[x].rewardshift[y] - 1 < 0)
                            actors[x].paramOne[y] += (reward - (reward - punishment) * (1
 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));
                        else
                            actors[x].paramOne[y] += (reward + (temptation - reward) * (1
 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                    }
                    else if (actors[x].interactions[a][y] == 2)
                        actors[x].paramOne[y] += sucker;
                    else if (actors[x].interactions[a][y] == 3)
                        actors[x].paramOne[y] += temptation;
                    else
                        actors[x].paramOne[y] += punishment;            // works as K in
Axelrod's second tournament
                    if (actors[x].encounters[y] <= 5)
                    {
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                        {
                            actors[x].variables[y][10 + (int)(Math.Pow(actors[x].
defections[y] + 1, 3)) % 5] = 1;
                            actors[x].defections[y]++;                      // works as
IPO1 in Axelrod's second tournament
                            return false;
                        }
                        else
                        {
```

```
                                actors[x].variables[y][10 + (int)(Math.Pow(actors[x].
defections[y] + 1, 3)) % 5] = 0;
                                return true;
                            }
                        }
                        if ((actors[x].encounters[y] - 1) % 5 == 0)
                        {
                            if (actors[x].array[y][0, 0] > actors[x].paramOne[y] - actors[x].
paramTwo[y])
                            {
                                actors[x].standing[y] = -actors[x].standing[y]; // works as
ICHAN in Axelrod's second tournament
                                actors[x].defections[y] += actors[x].standing[y];
                            }
                            actors[x].array[y][0, 0] = actors[x].paramOne[y] - actors[x].
paramTwo[y];        // works as KOLD in Axelrod's second tournament
                            actors[x].paramTwo[y] = actors[x].paramOne[y];        // works as
 KLAST in Axelrod's second tournament
                            if (actors[x].defections[y] < 0 || actors[x].defections[y] > 4)
                            {
                                if (actors[x].defections[y] < 0)
                                    actors[x].defections[y] = -1;
                                if (actors[x].defections[y] > 4)
                                    actors[x].defections[y] = 5;
                                if (actors[x].variables[y][10 + (int)(actors[x].encounters[y]
 - 1) % 5] == 1)
                                    return false;
                                return true;
                            }
                            actors[x].variables[y][10 + (int)(Math.Pow(actors[x].defections
[y] + 1, 3)) % 5] += actors[x].standing[y];
                            actors[x].defections[y] += actors[x].standing[y];
                        }
                        if (actors[x].variables[y][10 + (int)(actors[x].encounters[y] - 1) %
5] == 1)
                            return false;
                        return true;
                    }
                }
                return true;
            }
            else if (strategyNo == 53) //SMOODY RIK
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if ((actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y] ==
 3) && rnd.NextDouble() >= 0.9)
                        return false;
                    else if (actors[x].interactions[a][y] != 0)
                        return true;
                }
                if (rnd.NextDouble() >= 0.9)        //in the first move, according to Axelrod
's 2nd tournament, opponents previous move is cooperation.
                    return false;
                return true;
            }
            else if (strategyNo == 54) //FELD
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].interactions[a][y] == 1)
                        {
                            if (actors[x].rewardshift[y] - 1 < 0)
                                actors[x].paramOne[y] += (reward - (reward - punishment) * (1
 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));
                            else
                                actors[x].paramOne[y] += (reward + (temptation - reward) * (1
```

```
 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                    }
                    else if (actors[x].interactions[a][y] == 2)
                        actors[x].paramOne[y] += sucker;
                    else if (actors[x].interactions[a][y] == 3)
                        actors[x].paramOne[y] += temptation;
                    else
                        actors[x].paramOne[y] += punishment;                    // works ↙
as ISCORE in Axelrod's second tournament
                    actors[x].defections[y] = 1;                                // works ↙
as JPICK in Axelrod's second tournament
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]↙
 == 4)
                        actors[x].defections[y] = 0;
                    if (actors[x].encounters[y] >= 20)
                    {
                        actors[x].variables[y][(int)actors[x].standing[y]] = actors[x].  ↙
paramOne[y] - actors[x].paramTwo[y];
                        if (actors[x].standing[y] != 14)
                        {
                            if (actors[x].variables[y][(int)actors[x].standing[y] + 1] <=↙
 actors[x].variables[y][(int)actors[x].standing[y]])
                            {
                                if (actors[x].standing[y] != 10 && (actors[x].variables  ↙
[y][(int)actors[x].standing[y] - 1] > actors[x].variables[y][(int)actors[x].standing     ↙
[y]]))
                                {
                                    actors[x].standing[y]--;
                                    actors[x].defections[y] = 1;
                                }
                            }
                            else
                                actors[x].standing[y]++;
                        }
                        else
                        {
                            if (actors[x].variables[y][(int)actors[x].standing[y] - 1] > ↙
actors[x].variables[y][(int)actors[x].standing[y]])
                            {
                                actors[x].standing[y]--;
                                actors[x].defections[y] = 1;
                            }
                        }
                        actors[x].paramTwo[y] = actors[x].paramOne[y];          // works ↙
as KI in Axelrod's second tournament
                        actors[x].encounters[y] = 0;
                    }
                    actors[x].encounters[y]++;
                    if (actors[x].standing[y] == 10)
                        return true;
                    else if (actors[x].standing[y] == 11)
                    {
                        if (actors[x].defections[y] == 1)
                            return true;
                        if (rnd.NextDouble() <= 0.75)
                            return false;
                        return true;
                    }
                    else if (actors[x].standing[y] == 12)
                    {
                        if (actors[x].defections[y] == 1)
                            return true;
                        return false;
                    }
                    else if (actors[x].standing[y] == 13)
                    {
                        if (actors[x].defections[y] == 0)
                            return false;
                        if (rnd.NextDouble() <= 0.75)
```

```
                                return true;
                            return false;
                        }
                        else
                            return false;
                    }
                }
                actors[x].standing[y] = 12;              // works as JSTR in Axelrod's second
 tournament
                actors[x].encounters[y] = 1;             // works as KTRY in Axelrod's second
 tournament
                actors[x].variables[y][10] = 100;        // works as KEXP in Axelrod's second
 tournament
                actors[x].variables[y][11] = 100;
                actors[x].variables[y][12] = 100;
                actors[x].variables[y][13] = 100;
                actors[x].variables[y][14] = 100;
                return true;
            }
            else if (strategyNo == 55) //SNODGRASS
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].interactions[a][y] == 1)
                        {
                            if (actors[x].rewardshift[y] - 1 < 0)
                                actors[x].array[y][1, 1] += (reward - (reward - punishment) *
 (1 - Math.Pow(payoffshift, Math.Abs(actors[x].rewardshift[y] - 1))));     // actors[x].
defections[y] works as K in Axelrod's code (score of player x gained from interactions
with actor y so far)
                            else if (actors[x].rewardshift[y] - 1 >= 0)
                                actors[x].array[y][1, 1] += (reward + (temptation - reward) *
 (1 - Math.Pow(payoffshift, actors[x].rewardshift[y] - 1)));
                        }
                        else if (actors[x].interactions[a][y] == 2)
                            actors[x].array[y][1, 1] += sucker;
                        else if (actors[x].interactions[a][y] == 3)
                            actors[x].array[y][1, 1] += temptation;
                        else
                            actors[x].array[y][1, 1] += punishment;          // works as K in
 Axelrod's second tournament
                        do
                        {
                            do
                            {
                                actors[x].array[y][0, 0] = (int)(actors[x].paramOne[y] / 10);
       // works as CODE in Axelrod's second tournament
                                if (actors[x].paramOne[y] % 10 == 0)
                                    actors[x].variables[y][15 + (int)actors[x].array[y][0,
0]] = actors[x].array[y][1, 1];       // works as SC(CODE) in Axelrod's second tournament
; CODE = 15 + actors[x].array[y][0, 0]
                                if (actors[x].variables[y][9 + (int)actors[x].array[y][0, 0]]
 != 1)
                                    actors[x].paramOne[y] += 10;
                            }
                            while (actors[x].variables[y][9 + (int)actors[x].array[y][0, 0]]
!= 1);
                            actors[x].paramOne[y]++;
                            actors[x].variables[y][21 + (int)actors[x].array[y][0, 0]]++;
                            if (actors[x].array[y][0, 0] == 1)
                                return true;
                            if (actors[x].array[y][0, 0] == 2)
                                return false;
                            if (actors[x].array[y][0, 0] == 3)
                            {
                                if (actors[x].standing[y] == 1)
                                {
```

```
                                actors[x].standing[y] = 0;
                                return true;
                            }
                            actors[x].standing[y] = 1;
                            return false;
                        }
                        if (actors[x].array[y][0, 0] == 4)
                        {
                            if (actors[x].interactions[a][y] == 2 || actors[x].
interactions[a][y] == 4)
                                return false;
                            return true;
                        }
                        if (actors[x].array[y][0, 0] == 5)
                        {
                            if (actors[x].defections[y] == 1 && (actors[x].interactions
[a][y] == 2 || actors[x].interactions[a][y] == 4))
                                return false;
                            actors[x].defections[y] = 0;          // works as penultimate
 opponent's move
                            if (actors[x].interactions[a][y] == 2 || actors[x].
interactions[a][y] == 4)
                                actors[x].defections[y] = 1;
                            return true;
                        }
                        actors[x].array[y][1, 0] = 0;  // works as SGT in Axelrod's
second tournament, i.e. sum of gains of all active rules in their last 10 moves
                        for (int u = 1; u < 6; u++)
                        {
                            actors[x].array[y][1, 0] += (actors[x].variables[y][15 + u +
1] - actors[x].variables[y][15 + u]);
                            actors[x].variables[y][27 + u] += (actors[x].variables[y][15
+ u + 1] - actors[x].variables[y][15 + u]);    // works as GT(I) in Axelrod's second
tournament; I = [27 + u], i.e. sum of gains from all decisions based on a given rule (1-
5)
                        }
                        actors[x].array[y][0, 1] = (int)((9 * (int)(actors[x].array[y][1,
 0] / actors[x].paramTwo[y])) / 10);    // works as AMEAN in Axelrod's second tournament,
 i.e. 90% of the average sum of gains from the last 10 moves of all active rule
                        actors[x].paramTwo[y] = 0;
                        for (int f = 1; f < 6; f++)
                        {
                            if (actors[x].variables[y][9 + f] != 1)
                            {
                                if ((int)(10 * actors[x].variables[y][27 + f] / actors[x]
.variables[y][21 + f]) > actors[x].array[y][0, 1])
                                    actors[x].variables[y][9 + f] = 1;
                            }
                            else
                            {
                                if ((actors[x].variables[y][15 + f + 1] - actors[x].
variables[y][15 + f]) < actors[x].array[y][0, 1])
                                    actors[x].variables[y][9 + f] = 0;
                            }
                            if (actors[x].variables[y][9 + f] == 1)
                                actors[x].paramTwo[y]++;
                        }
                        actors[x].paramOne[y] = 10;
                    }
                    while (actors[x].array[y][0, 0] == 6);
                }
            }
            for (int v = 10; v < 16; v++)
                actors[x].variables[y][v] = 1;      // works as SL(I) in Axelrod's second
 tournament; v = 9 + I
            actors[x].variables[y][22] = 1;         // works as TM{I} in Axelrod's second
 tournament; x = 21 + I
            actors[x].paramOne[y] = 11;             // works as CN in Axelrod's second
tournament
```

```
            actors[x].paramTwo[y] = 5;              // works as CSRC in Axelrod's second tournament
            actors[x].standing[y] = 1;              // works as MYLM in Axelrod's second tournament, i.e. my last move
            return true;
        }
        else if (strategyNo == 56) //DUISMAN change from cooperation on odd moves to cooperation on odd interactions
        {
            actors[x].encounters[y]++;
            if ((int)actors[x].encounters[y] % 2 > 0)
                return true;
            else
                return false;
        }
        else if (strategyNo == 57) //ROBERTSON
        {
            actors[x].encounters[y]++;
            if (actors[x].encounters[y] == 20)
                actors[x].standing[y] = 0.1;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] == 4)
                {
                    actors[x].defections[y] = 0;            // works as COOPS in Axelrod's second tournament
                    if (actors[x].encounters[y] > 4)
                    {
                        actors[x].array[y][0, 1]++;         // works as STDEF in Axelrod's second tournament
                        actors[x].array[y][0, 0]++;         // works as OPDEF in Axelrod's second tournament
                        if (actors[x].paramTwo[y] != 0)     // works as MYDEF in Axelrod's second tournament, .FALSE. is [0]
                            actors[x].paramOne[y] = 1;      // works as OKDEF in Axelrod's second tournament, .TRUE. is [0]
                        actors[x].paramTwo[y] = 0;
                        if ((actors[x].array[y][0, 0] > actors[x].encounters[y] * actors[x].standing[y]) || actors[x].array[y][0, 1] > 2)
                            return false;
                        return true;
                    }
                    return false;
                }
                else if (actors[x].interactions[a][y] != 0)
                {
                    actors[x].array[y][0, 1] = 0;
                    actors[x].defections[y]++;
                    if (actors[x].array[y][0, 0] > actors[x].encounters[y] * actors[x].standing[y])
                    {
                        if (20 * actors[x].array[y][0, 0] <= actors[x].encounters[y] * actors[x].defections[y])
                            return true;
                        actors[x].paramTwo[y] = 0;
                        return false;
                    }
                    if ((int)actors[x].encounters[y] % (int)actors[x].array[y][1, 1] == 0 && actors[x].paramOne[y] == 0)
                    {
                        actors[x].paramTwo[y] = 1;
                        actors[x].array[y][1, 0]++;         // works as NODEF in Axelrod's second tournament
                        if ((int)actors[x].array[y][1, 0] % 6 == 0)
                            actors[x].array[y][1, 1]--;
                        if (actors[x].array[y][1, 1] < 1)
                            actors[x].array[y][1, 1] = 1;
                        return false;
```

```
                    }
                    actors[x].paramTwo[y] = 0;
                    return true;
                }
            }
            actors[x].standing[y] = 0.2;                    // works as DL in Axelrod's second tournament
            actors[x].array[y][1, 1] = 12;                  // works as ND in Axelrod's second tournament
            return true;
        }
        else if (strategyNo == 58) //RABBIE
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    actors[x].paramTwo[y] = actors[x].paramOne[y];      // works as LAST2 in Axelrod's second tournament
                    actors[x].paramOne[y] = 1;
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y] == 2)
                        actors[x].paramOne[y] = 0;
                    if (actors[x].encounters[y] > 2)
                    {
                        if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y] == 3)
                            actors[x].variables[y][9 + (int)actors[x].defections[y]]++;
                        // [10] - [13] works as COOP in Axelrod's second tournament
                        actors[x].variables[y][13 + (int)actors[x].defections[y]]++;
                        // [14] - [17] works as COUNT in Axelrod's second tournament
                        actors[x].variables[y][17 + (int)actors[x].defections[y]] = actors[x].variables[y][9 + (int)actors[x].defections[y]] / actors[x].variables[y][13 + (int)actors[x].defections[y]]; // [18] - [21] works as P in Axelrod's second tournament
                    }
                    actors[x].defections[y] = 2 * actors[x].paramTwo[y] + actors[x].paramOne[y] + 1;        // works as INDEX in Axelrod's second tournament
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] == 3)
                        actors[x].standing[y] = 0;
                    if (actors[x].encounters[y] > 22)
                    {
                        if (actors[x].standing[y] == 1)
                            return true;
                        double best = 0;
                        double[] con = { 0, 4, 6, 6, 8, 12 };       // works as CONST in Axelrod's second tournament
                        double[,] coef = { { 36, 0, 0, 0 }, { 16, 12, 12, 0 }, { 0, 18, 24, 0 }, { 12, 12, 9, 9 }, { 0, 16, 16, 12 }, { 0, 0, 0, 48 } };      // works as COEFF in Axelrod's second tournament
                        for (int q = 0; q < 6; q++)
                        {
                            double sum = con[q];
                            for (int r = 0; r < 4; r++)
                                sum += actors[x].variables[y][18 + r] * coef[q, r];
                            if (sum > best)
                            {
                                best = sum;
                                actors[x].variables[y][22] = q;         // works as IPOL in Axelrod's second tournament
                            }
                        }
                        if (actors[x].variables[y][22] == 5)
                            return false;
                        else if (actors[x].variables[y][22] == 4)
                        {
                            if (actors[x].defections[y] == 1 || actors[x].defections[y] == 2 || actors[x].defections[y] == 3)
```

```
                                return false;
                            }
                            else if (actors[x].variables[y][22] == 3)
                            {
                                if (actors[x].defections[y] == 1 || actors[x].defections[y] =↙
= 2)
                                    return false;
                            }
                            else if (actors[x].variables[y][22] == 2)
                            {
                                if (actors[x].defections[y] == 1 || actors[x].defections[y] =↙
= 3)
                                    return false;
                            }
                            else if (actors[x].variables[y][22] == 1)
                            {
                                if (actors[x].defections[y] == 1)
                                    return false;
                            }
                            return true;
                        }
                        if (actors[x].defections[y] == 1 || actors[x].defections[y] == 2)
                            return false;
                        return true;
                    }
                }
                actors[x].paramOne[y] = 1;                                  // works as LAST1 ↙
in Axelrod's second tournament
                return true;
            }
            else if (strategyNo == 59) //HALL
            {
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].encounters[y] == 2)
                        {
                            if (actors[x].interactions[a][y] == 2 || actors[x].interactions  ↙
[a][y] == 4)
                                return true;
                            return false;
                        }
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y]↙
 == 4)
                        {
                            actors[x].paramTwo[y]++;
                            if (actors[x].paramTwo[y] != 2)
                                return true;
                            actors[x].paramTwo[y] = 0;
                            return false;
                        }
                        actors[x].paramOne[y]++;
                        if (actors[x].paramOne[y] != 2)
                            return true;
                        actors[x].paramOne[y] = 0;
                        return false;
                    }
                }
                return true;
            }
            else if (strategyNo == 60) //FRIEDLAND
            {
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
```

```
                    if (actors[x].variables[y][10] == 1)        // works as JR in Axelrod
's second tournament
                        return false;
                    actors[x].variables[y][16] = 1;             // works as J in Axelrod
's second tournament
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y]
 == 3)
                        actors[x].variables[y][16] = 0;
                    if (actors[x].encounters[y] > 2)
                    {
                        if (actors[x].variables[y][11] == 0)
                        {
                            if (actors[x].variables[y][16] == 0)
                                actors[x].array[y][0, 0]++;     // works as QCA in
Axelrod's second tournament
                            actors[x].array[y][0, 1]++;         // works as QNA in
Axelrod's second tournament
                            actors[x].paramOne[y] = actors[x].array[y][0, 0] / actors[x].
array[y][0, 1];
                            actors[x].array[y][0, 0] *= 0.8;
                            actors[x].array[y][0, 1] *= 0.8;
                        }
                        else
                        {
                            if (actors[x].variables[y][16] == 0)
                                actors[x].array[y][1, 0]++;     // works as QCB in
Axelrod's second tournament
                            actors[x].array[y][1, 1]++;         // works as QNB in
Axelrod's second tournament
                            actors[x].paramTwo[y] = actors[x].array[y][1, 0] / actors[x].
array[y][1, 1];
                            actors[x].array[y][1, 0] *= 0.8;
                            actors[x].array[y][1, 1] *= 0.8;
                        }
                    }
                    actors[x].variables[y][11] = 0;             // works as IOLD in
Axelrod's second tournament
                    if (actors[x].interactions[a][y] == 3 || actors[x].interactions[a][y]
 == 4)
                        actors[x].variables[y][11] = 1;
                    if (actors[x].encounters[y] == 37)
                    {
                        if (actors[x].variables[y][13] == 1 && actors[x].variables[y][14]
 == 0 && actors[x].variables[y][15] > 10 && actors[x].variables[y][15] < 26 && actors[x].
variables[y][12] < 26)
                        {
                            actors[x].variables[y][10] = 1;
                            return false;
                        }
                    }
                    else if (actors[x].encounters[y] < 37 && actors[x].encounters[y] !=
1)
                    {
                        if (actors[x].defections[y] == actors[x].variables[y][16])      /
/ works as JL in Axelrod's second tournament
                        {
                            actors[x].standing[y]++;                                    /
/ works as JSM in Axelrod's second tournament
                            if (actors[x].standing[y] >= 3)
                                actors[x].variables[y][13] = 1;                         /
/ works as JS4 in Axelrod's second tournament
                            if (actors[x].standing[y] >= 11)
                                actors[x].variables[y][14] = 1;                         /
/ works as JS11 in Axelrod's second tournament
                        }
                        else
                        {
                            actors[x].variables[y][12]++;                               /
/ works as JSW in Axelrod's second tournament
```

```
                                actors[x].standing[y] = 1;
                            }
                            actors[x].variables[y][15] += actors[x].variables[y][16];       /↙
/ works as JT in Axelrod's second tournament
                            actors[x].defections[y] = actors[x].variables[y][16];
                        }
                        double polc = 6 * actors[x].paramOne[y] - 8 * actors[x].paramTwo[y] -↙
 2;
                        double polalt = 4 * actors[x].paramOne[y] - 5 * actors[x].paramTwo[y]↙
 - 1;
                        if (polc >= 0)
                        {
                            if (polc >= polalt)
                                return true;
                            if (actors[x].variables[y][11] == 0)
                                return false;
                            return true;
                        }
                        if (polalt >= 0)
                        {
                            if (actors[x].variables[y][11] == 0)
                                return false;
                            return true;
                        }
                        return false;
                    }
                }
                actors[x].paramOne[y] = 1;                          // works as ALPHA in     ↙
Axelrod's second tournament
                actors[x].paramTwo[y] = 0.3;                        // works as BETA in      ↙
Axelrod's second tournament
                return true;
            }
            else if (strategyNo == 61) //RANDOM
            {
                if (rnd.NextDouble() > 0.5)
                    return true;
                else
                    return false;
            }
            else if (strategyNo == 62) //HOTZ
            {
                actors[x].encounters[y]++;
                double prob;
                if (actors[x].encounters[y] > 300)
                    return false;
                else if (actors[x].encounters[y] < 300 && actors[x].encounters[y] >= 200)
                    prob = 0.15;
                else if (actors[x].encounters[y] < 200 && actors[x].encounters[y] >= 100)
                    prob = 0.05;
                else
                    prob = 0.1;
                if (rnd.NextDouble() < prob)
                    return true;
                return false;
            }
            else if (strategyNo == 63) //WSLS
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] == 3 || actors[x].interactions[a][y] == ↙
2)
                        return false;
                    else if (actors[x].interactions[a][y] != 0)
                        return true;
                }
                return true;
            }
            else if (strategyNo == 64) //STFT
```

```
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y] ==
3)
                        return true;
                    else if (actors[x].interactions[a][y] != 0)
                        return false;
                }
                return false;
            }
            else if (strategyNo == 65) //GTFT
            {
                double currentReward;
                if (actors[x].rewardshift[y] < 0)
                    currentReward = reward - (reward - punishment) * (1 - Math.Pow
(payoffshift, Math.Abs(actors[x].rewardshift[y])));
                else
                    currentReward = reward + (temptation - reward) * (1 - Math.Pow
(payoffshift, actors[x].rewardshift[y]));
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] ==
4)
                    {
                        double genLevel;
                        if ((currentReward - punishment) / (temptation - punishment) < (2 *
currentReward - sucker - temptation) / (currentReward - sucker))
                            genLevel = (currentReward - punishment) / (temptation -
punishment);
                        else
                            genLevel = (2 * currentReward - sucker - temptation) /
(currentReward - sucker);
                        if (genLevel < rnd.NextDouble())
                            return false;
                        else
                            return true;
                    }
                    else if (actors[x].interactions[a][y] != 0)
                        return true;
                }
                return true;
            }
            else if (strategyNo == 66) //CTFT
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if ((actors[x].standing[y] == 1 && actors[x].interactions[a][y] == 3) ||
(actors[x].standing[y] == 3 && (actors[x].interactions[a][y] == 3 || actors[x].
interactions[a][y] == 4)))
                    {
                        actors[x].standing[y] = 3;
                        return true;
                    }
                    else if ((actors[x].standing[y] == 1 && actors[x].interactions[a][y] ==
2) || (actors[x].standing[y] == 2 && (actors[x].interactions[a][y] == 4 || actors[x].
interactions[a][y] == 2)))
                    {
                        actors[x].standing[y] = 2;
                        return false;
                    }
                    else if (actors[x].interactions[a][y] != 0)
                    {
                        actors[x].standing[y] = 1;
                        return true;
                    }
                }
                return true;
            }
```

```
        else if (strategyNo == 67) //REMORSE
        {
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if ((actors[x].interactions[a][y] == 3 && (actors[x].standing[y] == 1 ||
actors[x].standing[y] == 3)) || (actors[x].interactions[a][y] == 4 && actors[x].standing
[y] == 3))
                {
                    actors[x].standing[y] = 3;
                    return true;
                }
                else if ((actors[x].interactions[a][y] == 2 && (actors[x].standing[y] ==
1 || actors[x].standing[y] == 2)) || (actors[x].standing[y] == 2 && actors[x].
interactions[a][y] == 4))
                {
                    actors[x].standing[y] = 2;
                    return false;
                }
                else if ((actors[x].standing[y] == 3 && actors[x].interactions[a][y] ==
2) || (actors[x].standing[y] == 2 && actors[x].interactions[a][y] == 3))
                {
                    actors[x].standing[y] = 1;
                    return false;
                }
                else if (actors[x].interactions[a][y] != 0)
                {
                    actors[x].standing[y] = 1;
                    return true;
                }
            }
            return true;
        }
        else if (strategyNo == 68) //WALT
        {
            actors[x].encounters[y]++;
            for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
            {
                if (actors[x].interactions[a][y] != 0)
                {
                    if (actors[x].interactions[a][y] == 1)
                        actors[x].defections[y]++;      // counts mutually cooperative
outcomes
                    double cooperativeness = actors[x].defections[y] / (actors[x].
encounters[y] - 1);
                    double occurence = interaction(summa, actors[x].score, actors[y].
score, actors[x].row, actors[y].row, actors[x].column, actors[y].column);
                    if (occurence >= 0.67)
                    {
                        if (cooperativeness >= 0.76)
                        {
                            if (actors[x].interactions[a][y] == 2 || actors[x].
interactions[a][y] == 4)
                            {
                                if (rnd.NextDouble() > cooperativeness)
                                    return false;
                            }
                            return true;
                        }
                        else if (cooperativeness < 0.76 && cooperativeness >= 0.51) //TFT
                        {
                            if (actors[x].interactions[a][y] == 1 || actors[x].
interactions[a][y] == 3)
                                return true;
                            return false;
                        }
                        else
                        {
                            if (actors[x].interactions[a][y] == 1 || actors[x].
interactions[a][y] == 3)
```

```
                                {
                                    if (rnd.NextDouble() < cooperativeness)
                                        return true;
                                }
                                return false;
                            }
                        }
                        else if (occurence < 0.67 && occurence >= 0.34)
                        {
                            if (cooperativeness >= 0.76)
                            {
                                if (actors[x].interactions[a][y] == 2 || actors[x].
    interactions[a][y] == 4)
                                {
                                    if (rnd.NextDouble() > cooperativeness)
                                        return false;
                                }
                                return true;
                            }
                            else if (cooperativeness < 0.76 && cooperativeness >= 0.26) //TFT
                            {
                                if (actors[x].interactions[a][y] == 1 || actors[x].
    interactions[a][y] == 3)
                                    return true;
                                return false;
                            }
                            else
                            {
                                if (actors[x].interactions[a][y] == 1 || actors[x].
    interactions[a][y] == 3)
                                {
                                    if (rnd.NextDouble() < cooperativeness)
                                        return true;
                                }
                                return false;
                            }
                        }
                        else
                        {
                            if (cooperativeness >= 0.51)
                            {
                                if (actors[x].interactions[a][y] == 2 || actors[x].
    interactions[a][y] == 4)
                                {
                                    if (rnd.NextDouble() > cooperativeness)
                                        return false;
                                }
                                return true;
                            }
                            else if (cooperativeness < 0.51 && cooperativeness >= 0.26) //TFT
                            {
                                if (actors[x].interactions[a][y] == 2 || actors[x].
    interactions[a][y] == 4)
                                    return false;
                                return true;
                            }
                            else
                            {
                                if (actors[x].interactions[a][y] == 1 || actors[x].
    interactions[a][y] == 3)
                                {
                                    if (rnd.NextDouble() < cooperativeness)
                                        return true;
                                }
                                return false;
                            }
                        }
                    }
                }
```

```
                return true;
            }
            else if (strategyNo == 69) //BALANCE
            {
                double mean = summa / actors[x].encounters.GetLength(0);
                double variable = new double();
                if (actors[y].score >= mean)
                {
                    if (actors[y].score == 0 && mean == 0)
                        variable = 0.5;
                    else
                        variable = 1 - 0.5 * mean / actors[y].score;
                }
                else
                    variable = 0.5 * actors[y].score / mean;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (variable < 0.34)
                        {
                            if (actors[x].interactions[a][y] == 2 || actors[x].interactions ↙
[a][y] == 4)
                            {
                                if (rnd.NextDouble() < variable)
                                    return false;
                            }
                            return true;
                        }
                        else if (variable >= 0.34 && variable < 0.67)   //TFT
                        {
                            if (actors[x].interactions[a][y] == 2 || actors[x].interactions ↙
[a][y] == 4)
                                return false;
                            return true;
                        }
                        else
                        {
                            if (actors[x].interactions[a][y] == 1 || actors[x].interactions ↙
[a][y] == 3)
                            {
                                if (rnd.NextDouble() > variable)
                                    return true;
                            }
                            return false;
                        }
                    }
                }
                return true;

            }
            else if (strategyNo == 70) //BANDWAGON
            {
                double mean = summa / actors[x].encounters.GetLength(0);
                double variable = new double();
                if (actors[y].score >= mean)
                {
                    if (actors[y].score == 0 && mean == 0)
                        variable = 0.5;
                    else
                        variable = 1 - 0.5 * mean / actors[y].score;
                }
                else
                    variable = 0.5 * actors[y].score / mean;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (variable < 0.34)
```

```
                        {
                            if (actors[x].interactions[a][y] == 1 || actors[x].interactions
[a][y] == 3)
                            {
                                if (rnd.NextDouble() < variable)
                                    return true;
                            }
                            return false;
                        }
                        else if (variable >= 0.34 && variable < 0.67)   //TFT
                        {
                            if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                                return false;
                            return true;
                        }
                        else
                        {
                            if (actors[x].interactions[a][y] == 2 || actors[x].interactions
[a][y] == 4)
                            {
                                if (rnd.NextDouble() > variable)
                                    return false;
                            }
                            return true;
                        }
                    }
                }
                return true;

            }
            else if (strategyNo == 71) //WEAKLING
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if ((actors[x].interactions[a][y] == 1 && (actors[x].standing[y] == 1 ||
actors[x].standing[y] == 2 || actors[x].standing[y] == 3)) || (actors[x].interactions[a]
[y] == 2 && actors[x].standing[y] == 3) || (actors[x].interactions[a][y] == 3 && actors
[x].standing[y] == 2))
                    {
                        actors[x].standing[y] = 1;
                        return false;
                    }
                    else if ((actors[x].interactions[a][y] == 2 && (actors[x].standing[y] ==
1 || actors[x].standing[y] == 2)) || (actors[x].interactions[a][y] == 4 && actors[x].
standing[y] == 2))
                    {
                        actors[x].standing[y] = 2;
                        return false;
                    }
                    else if ((actors[x].interactions[a][y] == 3 && (actors[x].standing[y] ==
3 || actors[x].standing[y] == 1)) || (actors[x].interactions[a][y] == 4 && actors[x].
standing[y] == 3))
                    {
                        actors[x].standing[y] = 3;
                        return true;
                    }
                    else if (actors[x].interactions[a][y] != 0)
                        return true;
                }
                return false;
            }
            else if (strategyNo == 72) //GROFMAN FIRST TOURNAMENT
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] ==
3)
                    {
```

```
                        if (rnd.NextDouble() > 2.0 / 7.0)
                            return false;
                        return true;
                    }
                    else if (actors[x].interactions[a][y] != 0)
                        return true;
                }
                return true;
            }
            else if (strategyNo == 73) //FELD FIRST TOURNAMENT
            {
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] ==
4)
                        return false;
                    else if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].encounters[y] > 200)
                        {
                            if (rnd.NextDouble() <= 0.5)
                                return true;
                            return false;
                        }
                        if (rnd.NextDouble() <= 0.5 + (0.0025 * 200 - actors[x].encounters
[y]))
                            return true;
                        return false;
                    }
                }
                return true;
            }
            else if (strategyNo == 74) //JOSS FIRST TOURNAMENT
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] ==
4)
                        return false;
                    else if (actors[x].interactions[a][y] != 0)
                    {
                        if (rnd.NextDouble() > 0.9)
                            return false;
                        else
                            return true;
                    }
                }
                return true;
            }
            else if (strategyNo == 75) //SHUBIK
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].defections[y] > actors[x].encounters[y])
                        {
                            if (actors[x].interactions[a][y] == 3 || actors[x].interactions
[a][y] == 4)
                                actors[x].encounters[y]++;
                            if (actors[x].defections[y] == actors[x].encounters[y])
                                return true;
                            return false;
                        }
                        else if (actors[x].defections[y] == actors[x].encounters[y])
                        {
                            if (actors[x].interactions[a][y] == 2)
                            {
```

```
                                actors[x].defections[y]++;
                                actors[x].encounters[y] = 0;
                                return false;
                            }
                            return true;
                        }
                    }
                }
                return true;
            }
            else if (strategyNo == 76) //DAVIS
            {
                actors[x].encounters[y]++;
                if (actors[x].encounters[y] > 10)
                {
                    if (actors[x].defections[y] > 0)
                        return false;
                    for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                    {
                        if (actors[x].interactions[a][y] == 2 || actors[x].interactions[a][y] ↙
== 4)
                        {
                            actors[x].defections[y]++;
                            return false;
                        }
                        else if (actors[x].interactions[a][y] != 0)
                            return true;
                    }
                }
                return true;
            }
            else if (strategyNo == 77) // TULLOCK
            {
                actors[x].standing[y] = 0;
                actors[x].defections[y] = 0;
                actors[x].encounters[y]++;
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        actors[x].standing[y]++;
                        if (actors[x].interactions[a][y] == 1 || actors[x].interactions[a][y] ↙
== 3)
                            actors[x].defections[y]++;
                        if (actors[x].standing[y] == 10)
                            break;
                    }
                }
                if (actors[x].encounters[y] > 11)
                {
                    if (rnd.NextDouble() < 0.9 * actors[x].defections[y] / 10)
                        return true;
                    return false;
                }
                return true;
            }
            else if (strategyNo == 78) // ALLD
                return false;
            else if (strategyNo == 79) // ALLC
                return true;
            else //if (strategyNo == 80) // VIENNA
            {
                for (int a = actors[x].interactions.Count - 1; a >= 1; a--)
                {
                    if (actors[x].interactions[a][y] != 0)
                    {
                        if (actors[x].interactions[a][y] == 1)
                        {
                            if (rnd.NextDouble() > actors[x].standing[0])
```

```
                                return false;
                            return true;
                        }
                        else if (actors[x].interactions[a][y] == 2)
                        {
                            if (rnd.NextDouble() > actors[x].standing[1])
                                return false;
                            return true;
                        }
                        else if (actors[x].interactions[a][y] == 3)
                        {
                            if (rnd.NextDouble() > actors[x].standing[2])
                                return false;
                            return true;
                        }
                        else
                        {
                            if (rnd.NextDouble() > actors[x].standing[3])
                                return false;
                            return true;
                        }
                    }
                }
                if (rnd.NextDouble() > actors[x].standing[0])
                    return false;
                return true;
            }
        }
        private double interaction(double sum, double playerPowerA, double playerPowerB,
    double rowA, double rowB, double columnA, double columnB)
        {
            double distanceVar = new double();
            double powerVar = new double();
            if (power == true)
            {
                // power variable computation follows
                double powerVarA;
                double powerVarB;
                double avg = sum / (x * y);
                if (playerPowerA >= avg)
                {
                    if (playerPowerA == 0 && avg == 0)
                        powerVarA = 0.5;
                    else
                        powerVarA = 1 - 0.5 * (avg / playerPowerA);
                }
                else
                    powerVarA = 0.5 * (playerPowerA / avg);
                if (playerPowerB >= avg)
                {
                    if (playerPowerB == 0 && avg == 0)
                        powerVarB = 0.5;
                    else
                        powerVarB = 1 - 0.5 * (avg / playerPowerB);
                }
                else
                    powerVarB = 0.5 * (playerPowerB / avg);
                powerVar = (powerVarA + powerVarB) / 2;
            }
            if (geography == true)
            {
                double distance;
                // distance variable computation follows
                if (Math.Abs(rowA - rowB) <= ((double)x / 2) && Math.Abs(columnA - columnB)
    <= ((double)y / 2))
                    distance = Math.Sqrt(Math.Pow(rowA - rowB, 2) + Math.Pow(columnA -
    columnB, 2));
                else if (Math.Abs(rowA - rowB) <= ((double)x / 2) && Math.Abs(columnA -
    columnB) > ((double)y / 2))
```

```
                distance = Math.Sqrt(Math.Pow(rowA - rowB, 2) + Math.Pow(y - Math.Abs
(columnA - columnB), 2));
            else if (Math.Abs(rowA - rowB) > ((double)x / 2) && Math.Abs(columnA -
columnB) <= ((double)y / 2))
                distance = Math.Sqrt(Math.Pow(x - Math.Abs(rowA - rowB), 2) + Math.Pow
(columnA - columnB, 2));
            else
                distance = Math.Sqrt(Math.Pow(x - Math.Abs(rowA - rowB), 2) + Math.Pow(y
- Math.Abs(columnA - columnB), 2));
            distanceVar = (maxDistance - distance) / (maxDistance - 1);     // One (1) is
 minimum distance between of interacting actors - two bordering cells
        }
        // Occurence of interaction computation follows
        if (power == true && geography == true)
            return (powerVar + distanceVar) / 2;
        else if (power == true && geography == false)
            return powerVar;
        else if (power == false && geography == true)
            return distanceVar;
        else
            return 1;
    }
    private void mnuHelp_Click(object sender, EventArgs e)
    {
        round.Enabled = false;
        FormHelp formHelp = new FormHelp();
        formHelp.ShowDialog();
    }
    private void btnStart_Click(object sender, EventArgs e)
    {
        round.Enabled = !round.Enabled;     //Iterations
        btnWinner.Enabled = true;
    }
    private void round_Tick(object sender, EventArgs e)
    {
        number++;
        txtRound.Text = number.ToString();
        summa = 0;
        for (int o = 0; o < x * y; o++)
            actors[o].interaction = new int[x * y];
        gain = new double[x, y];
        for (int a = 0; a < x; a++)
        {
            for (int b = 0; b < y; b++)
            {
                gain[a, b] = 0;
                summa += input[a, b];
            }
        }
        int v = 0;
        for (int n = 0; n < x * y; n++)
        {
            for (int m = n + 1; m < x * y; m++)
            {
                if (interaction(summa, actors[n].score, actors[m].score, actors[n].row,
actors[m].row, actors[n].column, actors[m].column) >= rnd.NextDouble())
                {
                    if (rnd.NextDouble() <= misperception)
                        mispercieve(n, m);
                    strategyA = strategies(n, m);
                    if (rnd.NextDouble() <= misconduct)     //Misconduct
                        strategyA = !strategyA;
                    if (rnd.NextDouble() <= misperception)
                        mispercieve(m, n);
                    strategyB = strategies(m, n);
                    if (rnd.NextDouble() <= misconduct)     //Misconduct
                        strategyB = !strategyB;
                    action(n, m);
                    struc[v]++;
```

```
                    }
                    v++;
                }
            }
            for (int a = 0; a < x; a++)
            {
                for (int b = 0; b < y; b++)
                {
                    input[a, b] += gain[a, b];  // Adding gains from each round to power (sum↙
    of gains) of each player at the end of that round
                    Inputs.Rows[a][b] = input[a, b];
                }
            }
            dgvResults.DataSource = Inputs;
            outputs = new double[x * y]; //storing final power positions of all actors in all↙
    rounds after interactions
            int i = 0;
            for (int a = 0; a < x; a++)
            {
                for (int b = 0; b < y; b++)
                {
                    outputs[i] = input[a, b];
                    actors[i].score = input[a, b];
                    actors[i].interactions.Add(actors[i].interaction);
                    i += 1;
                }
            }
            outcomes.Add(outputs);
            if (number % 100 == 0)
            {
                int[] present = new int[x * y * (x * y - 1) / 2];
                for (int a = 0; a < x * y * (x * y - 1) / 2; a++)
                    present[a] = struc[a];
                structure.Add(present);
            }
            if (number == 10000)
            {
                round.Enabled = false;
                System.Console.Beep();
            }
        }
        private void btnReset_Click(object sender, EventArgs e)
        {
            round.Enabled = false;
            btnWinner.Enabled = false;
            Inputs.RejectChanges();
            dgvResults.DataSource = Inputs;
            coop = new int[x * y, x * y];
            input = new double[x, y];
            Cooperation.RejectChanges();
            Output.RejectChanges();
            Structures.RejectChanges();
            structure.Clear();
            outcomes.Clear();
            outputs = new double[x * y];
            int g = 0;
            struc = new int[x * y * (x * y - 1) / 2];
            for (int n = 0; n < x * y; n++)
            {
                actors[n].interactions.Clear();
                actors[n].interaction = new int[x * y];
                input[n / y, n % y] = Double.Parse(Inputs.Rows[n / y][n % y].ToString());
                actors[n].strategy = Int32.Parse(Strategies.Rows[n / y][n % y].ToString());
                if (actors[n].strategy == 80)
                {
                    if (actors[n].standing[0] == 0 && actors[n].standing[1] == 0 && actors[n]↙
    .standing[2] == 0)
                    {
                        if (actors[n].standing[3] == 1)
```

```
                        outputs[n] = 9999910;
                    else
                        outputs[n] = 9999990 + actors[n].standing[3] * 10;
                }
                else
                    outputs[n] = actors[n].standing[0] * 10000000 + actors[n].standing[1] ↙
* 100000 + actors[n].standing[2] * 1000 + actors[n].standing[3] * 10;
            }
            else
                outputs[n] = actors[n].strategy;
            actors[n].score = input[n / y, n % y];
            for (int m = 0; m < x * y; m++)
            {
                if (m > n)
                {
                    struc[g] = n + 1;
                    g++;
                }
                actors[n].rewardshift[m] = 0;
                actors[n].encounters[m] = 0;
                actors[n].defections[m] = 0;
                actors[n].standing[m] = 1;
                actors[n].paramOne[m] = 0;
                actors[n].paramTwo[m] = 0;
                coop[n, m] = 0;
                actors[n].interaction[m] = actors[m].strategy;
                actors[n].array[m] = new double[2, 2];
                actors[n].variables[m] = new double[100];
            }
            actors[n].interactions.Add(actors[n].interaction);
        }
        structure.Add(struc);
        g = 0;
        struc = new int[x * y * (x * y - 1) / 2];
        for (int p = 0; p < x * y; p++)
        {
            for (int r = p + 1; r < x * y; r++)
            {
                struc[g] = r + 1;
                g++;
            }
        }
        structure.Add(struc);
        struc = new int[x * y * (x * y - 1) / 2];
        txtRound.Text = "0";
        number = 0;
        outcomes.Add(outputs);
    }
    private void btnVerify_Click(object sender, EventArgs e)
    {
        round.Enabled = false;
        FormExamination formExamination = new FormExamination();
        formExamination.misconductLevel = misconduct;
        formExamination.misperceptionLevel = misperception;
        formExamination.rows = x;
        formExamination.columns = y;
        formExamination.table = Inputs;
        formExamination.ShowDialog();
    }
    private void FormSimulation_FormClosed(object sender, FormClosedEventArgs e)
    {
        round.Enabled = false;
    }
    private void btnWinner_Click(object sender, EventArgs e)
    {
        FormStrategies formStrategies = new FormStrategies();
        round.Enabled = false;
        double max = 0;
        int rowMax = new int();
```

```
            int colMax = new int();
            int strMax = new int();

            double tempMax = 1e26;
            int n = 0;
            List<string> list = new List<string>();
            for (int i = 1; i <= x * y; i++)
                list.Add("");
            for (int i = 1; i <= x * y; i++)
            {
                max = 0;
                for (int a = 0; a < x; a++)
                {
                    for (int b = 0; b < y; b++)
                    {
                        if (input[a, b] > max && input[a, b] < tempMax)
                        {
                            n = i;
                            max = input[a, b];
                            rowMax = a + 1;
                            colMax = b + 1;
                            strMax = Int32.Parse(Strategies.Rows[a][b].ToString());
                            formStrategies.determine(strMax);
                            if (strMax == 80)
                                list[i - 1] = +i + ". place is strategy " + formStrategies.
name + " (" + actors[a * y + b].standing[0] + "; " + actors[a * y + b].standing[1] + "; "
 + actors[a * y + b].standing[2] + "; " + actors[a * y + b].standing[3] + ") after (CC;
CD; DC; DD). Row " + rowMax + ". Column " + colMax + ". It gained " + Math.Round(max) + "
 points.";
                            else
                                list[i - 1] = +i + ". place is strategy " + formStrategies.
name + ". Row " + rowMax + ". Column " + colMax + ". It gained " + Math.Round(max) + "
points.";
                        }
                        else if (input[a, b] == max)
                        {
                            n++;
                            rowMax = a + 1;
                            colMax = b + 1;
                            strMax = Int32.Parse(Strategies.Rows[a][b].ToString());
                            formStrategies.determine(strMax);
                            if (strMax == 80)
                                list[n - 1] = +i + ". place is strategy " + formStrategies.
name + " (" + actors[a * y + b].standing[0] + "; " + actors[a * y + b].standing[1] + "; "
 + actors[a * y + b].standing[2] + "; " + actors[a * y + b].standing[3] + ") after (CC;
CD; DC; DD). Row " + rowMax + ". Column " + colMax + ". It gained " + Math.Round(max) + "
 points.";
                            else
                                list[n - 1] = +i + ". place is strategy " + formStrategies.
name + ". Row " + rowMax + ". Column " + colMax + ". It gained " + Math.Round(max) + "
points.";
                        }
                    }
                }
                tempMax = max;
                i = n;
            }
            FormResults formResults = new FormResults();
            formResults.results = list;
            formResults.ShowDialog();
        }
        private void gainsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            round.Enabled = false;
            for (int n = 0; n < outcomes.Count; n++)
            {
                Output.Rows.Add();
                for (int m = 0; m < x * y; m++)
                    Output.Rows[n][m] = Math.Round(outcomes[n][m]);
```

```
            }
            SaveFileDialog saveGains = new SaveFileDialog();
            saveGains.Filter = "Xml files (*.xml)|*.xml";
            saveGains.Title = "Save Gains";
            if (saveGains.ShowDialog() == DialogResult.OK)
            {
                Output.TableName = "gains";
                this.Output.WriteXml(saveGains.FileName.ToString());
            }
        }
        private void cooperationToolStripMenuItem_Click(object sender, EventArgs e)
        {
            round.Enabled = false;
            for (int n = 0; n < x * y; n++)
            {
                for (int m = 0; m < x * y; m++)
                    Cooperation.Rows[n][m] = coop[n, m];
            }
            SaveFileDialog saveCoop = new SaveFileDialog();
            saveCoop.Filter = "Xml files (*.xml)|*.xml";
            saveCoop.Title = "Save Cooperations";
            if (saveCoop.ShowDialog() == DialogResult.OK)
            {
                Cooperation.TableName = "coop";
                this.Cooperation.WriteXml(saveCoop.FileName.ToString());
            }
        }
        private void interactionsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            round.Enabled = false;
            for (int n = 0; n < structure.Count; n++)
                Structures.Columns.Add();
            for (int m = 0; m < x * y * (x * y - 1) / 2; m++)
            {
                Structures.Rows.Add();
                for (int d = 0; d < structure.Count; d++)
                    Structures.Rows[m][d] = structure[d][m];
            }
            SaveFileDialog saveInter = new SaveFileDialog();
            saveInter.Filter = "Xml files (*.xml)|*.xml";
            saveInter.Title = "Save Emergent Structure";
            if (saveInter.ShowDialog() == DialogResult.OK)
            {
                Structures.TableName = "inter";
                this.Structures.WriteXml(saveInter.FileName.ToString());
            }
        }
        private void distributionToolStripMenuItem_Click(object sender, EventArgs e)
        {
            round.Enabled = false;
            DataTable Tableau = new DataTable();
            for (int m = 0; m < y; m++)
                Tableau.Columns.Add();
            for (int m = 0; m < x; m++)
                Tableau.Rows.Add();
            for (int n = 0; n < x * y; n++)
                Tableau.Rows[n / y][n % y] = outcomes[0][n];
            SaveFileDialog saveDist = new SaveFileDialog();
            saveDist.Filter = "Xml files (*.xml)|*.xml";
            saveDist.Title = "Save Distribution";
            if (saveDist.ShowDialog() == DialogResult.OK)
            {
                Tableau.TableName = "dist";
                Tableau.WriteXml(saveDist.FileName.ToString());
            }
        }
    }
}
```