**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. Н.Э. БАУМАНА**

Факультет: Информатика и системы управления Кафедра: Информационная безопасность (ИУ8)

**ТЕОРИЯ ИГР И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ**

**Лабораторная работа № 4 на тему:**

«Критерии выбора оптимальных стратегий в неантагонистических играх и свойства оптимальных решений»

Вариант 4

**Преподаватель:**

Коннова Н.С.

**Студент**:

Куликова А.В.

**Группа:**

ИУ8-21М

**Цель работы**

Изучить критерии выбора стратегий в неантагонистической бескоалиционной игре двух игроков на основе равновесия Нэша и оптимальности по Парето. Проверить данные критерии на примере известных игр. Исследовать свойства оптимальных решений неантагонистических бескоалиционных игр на примере биматричных (2 х 2)-игр.

**Постановка задачи**

1) Сгенерировать случайную биматричную игру (10х10). Найти ситуации, равновесные по Нэшу и оптимальные по Парето, пересечение данных множеств. Произвести проверку реализованных алгоритмов на примере трёх известных игр: **семейный спор, перекрёсток со смещением, дилемма заключённого**.

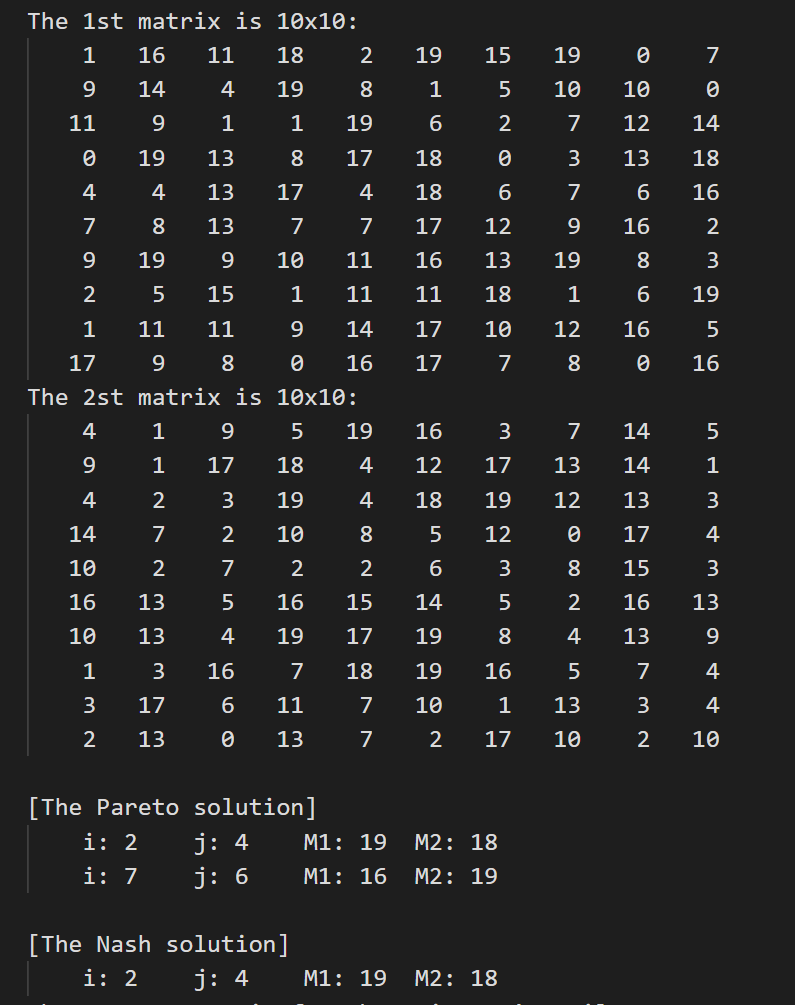
2) Для заданной по варианту Г(𝐴,𝐵) – биматричной (2 х 2)-игры, пользуясь теоремами о свойствах оптимальных решений, найдите ситуации, равновесные по Нэшу, для исходной игры и для ее смешанного расширения..

**Ход работы**

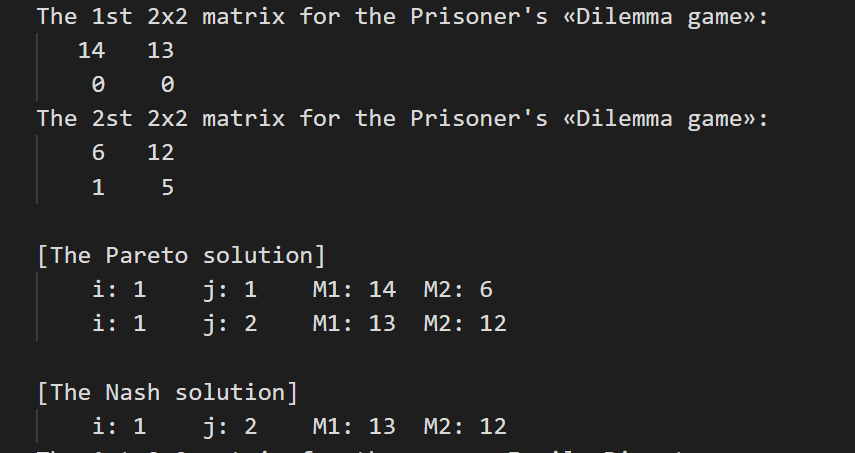
Ниже приведены биматричные игры для второй части задания. Первая часть задания подразумевает генерацию случайной платежной матрицы.

Биматрическая игра:

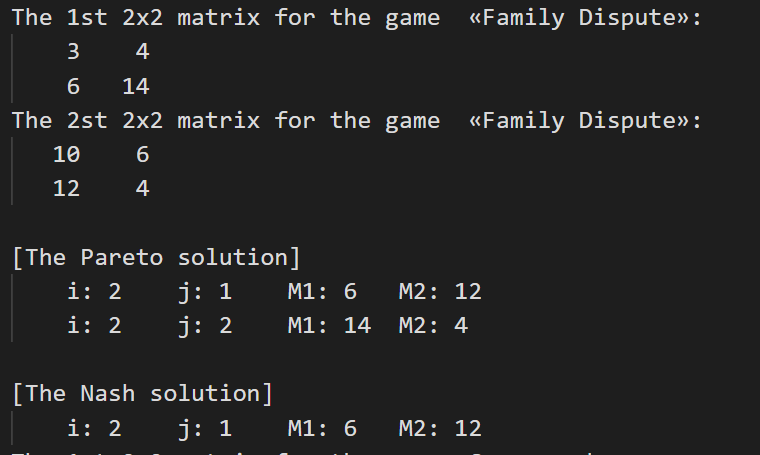
Случайная биматричная игра:



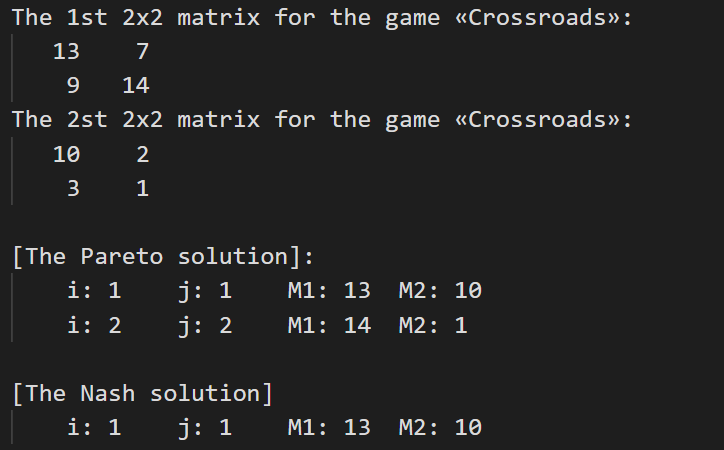
**матрица для игры «Дилемма заключенного»:**



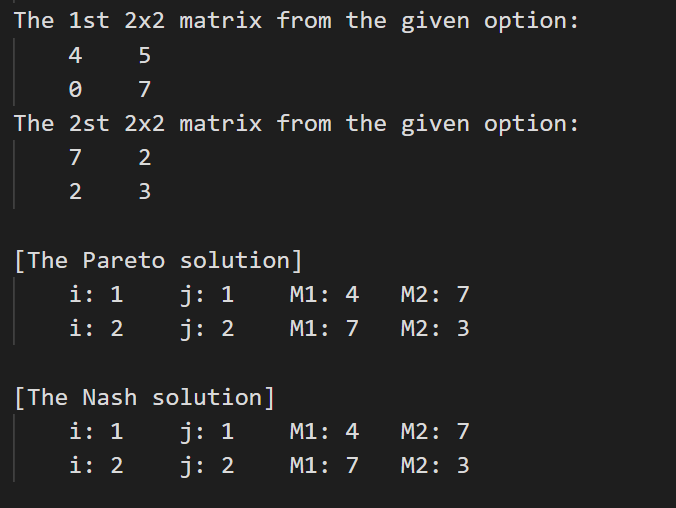
**матрица для игры «Семейный спор»:**



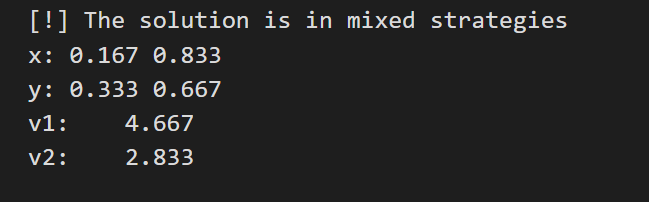
**матрица для игры «Перекресток»:**



По заданию:



Как видим, у этой игры есть две равновесные по Нэшу ситуации в чистых стратегиях, поэтому в смешанном дополнении игры существует ещё одна вполне смешанная ситуация равновесия, которую мы можем рассчитать по формулам из методических указаний к заданию:



Результат программы, представленной в приложении А.

**Выводы**

В ходе проделанной работы было найдено решение по Нэшу и Парето, а также исследован ряд теоретических моментов.

Один игрок имеет строго доминирующую стратегию, существует только одно решение по Нэшу.

**Контрольные вопросы**

1. ***Определение равновесных ситуаций в игре.***

Равновесные ситуации в игре — это такие ситуации, при которых ни одна из сторон не имеет мотивации изменить свои действия, так как любое изменение приведет к ухудшению их положения. В равновесной ситуации игроки принимают оптимальные для себя решения, учитывая действия других игроков.

1. ***Оптимальность по Парето в бескоалиционных неантагонистических играх нескольких игроков?***

Оптимальность по Парето в бескоалиционных неантагонистических играх нескольких игроков означает, что существует такое распределение выигрышей между игроками, при котором ни один игрок не может улучшить свою позицию без ухудшения позиции других игроков. Другими словами, оптимальность по Парето говорит о том, что в данной игровой ситуации нет возможности улучшить положение одного игрока без ущемления интересов других игроков.

В бескоалиционных неантагонистических играх нескольких игроков оптимальность по Парето может быть достигнута через рациональное выбор стратегий каждым игроком, учитывая действия остальных участников игры. Если все игроки выбирают оптимальные для себя стратегии, то возможно достижение равновесия по Парето, когда никому не выгодно менять свою стратегию.

1. ***Нахождение равновесия в смешанных стратегиях?***

Равновесие в смешанных стратегиях - это концепция в теории игр, при которой каждый игрок выбирает свою стратегию с определенной вероятностью, таким образом, что ни один игрок не имеет мотивации изменить свою стратегию при условии, что остальные игроки действуют согласно своим вероятностям.

Для нахождения равновесия в смешанных стратегиях в игре нужно выполнить следующие шаги

1. Сначала определить все возможные стратегии для каждого игрока.

2. Затем определить вероятности, с которыми каждый игрок будет выбирать каждую из своих стратегий.

3. Посчитать ожидаемый выигрыш каждого игрока при данном распределении вероятностей.

4. Проверить, есть ли у какого-либо игрока мотивация изменить свою стратегию, чтобы увеличить свой выигрыш.

Если ни у одного игрока нет мотивации изменить свою стратегию, то найденное распределение вероятностей является равновесием в смешанных стратегиях.

***4. Пусть Г(𝐴,𝐵) – биматричная (𝑚 х 𝑚)-игра. Всегда ли существуют смешанные стратегии 𝑥∗, 𝑦∗ игроков 1 и 2 соответственно, такие что (𝑥∗, 𝑦∗) – ситуация равновесия по Нэшу?***

Для биматричных игр с размером матрицы m × m не всегда существуют смешанные стратегии игроков 1 и 2, которые образуют равновесие по Нэшу. В некоторых случаях равновесие по Нэшу в смешанных стратегиях может не существовать.

Например, в игре камень-ножницы-бумага, где матрица выигрышей имеет размер 3 × 3, равновесие по Нэшу в смешанных стратегиях не существует. В этой игре каждый игрок имеет по три стратегии (камень, ножницы, бумага), и ни один игрок не имеет мотивации изменить свою стратегию при условии, что другой игрок действует случайно. Таким образом, равновесие по Нэшу в этой игре возможно только в чистых стратегиях.

Тем не менее, для других биматричных игр с размером матрицы m × m равновесие по Нэшу в смешанных стратегиях может существовать. Для таких игр можно применить методы нахождения равновесия в смешанных стратегиях, описанные ранее.

**Приложение А**

\*\*\*\*GENERATION START\*\*\*\*

[Dilemma game]

M1\_1:

14 13

0 0

M2\_1:

6 12

1 5

[Family Dispute]

M1\_2:

3 4

6 14

M2\_2:

10 6

12 4

[Crossroads]

M1\_3:

13 7

9 14

M2\_3:

10 2

3 1

\*\*\*\*GENERATION END\*\*\*\*

The 1st matrix is 10x10:

1 16 11 18 2 19 15 19 0 7

9 14 4 19 8 1 5 10 10 0

11 9 1 1 19 6 2 7 12 14

0 19 13 8 17 18 0 3 13 18

4 4 13 17 4 18 6 7 6 16

7 8 13 7 7 17 12 9 16 2

9 19 9 10 11 16 13 19 8 3

2 5 15 1 11 11 18 1 6 19

1 11 11 9 14 17 10 12 16 5

17 9 8 0 16 17 7 8 0 16

The 2st matrix is 10x10:

4 1 9 5 19 16 3 7 14 5

9 1 17 18 4 12 17 13 14 1

4 2 3 19 4 18 19 12 13 3

14 7 2 10 8 5 12 0 17 4

10 2 7 2 2 6 3 8 15 3

16 13 5 16 15 14 5 2 16 13

10 13 4 19 17 19 8 4 13 9

1 3 16 7 18 19 16 5 7 4

3 17 6 11 7 10 1 13 3 4

2 13 0 13 7 2 17 10 2 10

[The Pareto solution]

i: 2 j: 4 M1: 19 M2: 18

i: 7 j: 6 M1: 16 M2: 19

[The Nash solution]

i: 2 j: 4 M1: 19 M2: 18

The 1st 2x2 matrix for the Prisoner's «Dilemma game»:

14 13

0 0

The 2st 2x2 matrix for the Prisoner's «Dilemma game»:

6 12

1 5

[The Pareto solution]

i: 1 j: 1 M1: 14 M2: 6

i: 1 j: 2 M1: 13 M2: 12

[The Nash solution]

i: 1 j: 2 M1: 13 M2: 12

The 1st 2x2 matrix for the game «Family Dispute»:

3 4

6 14

The 2st 2x2 matrix for the game «Family Dispute»:

10 6

12 4

[The Pareto solution]

i: 2 j: 1 M1: 6 M2: 12

i: 2 j: 2 M1: 14 M2: 4

[The Nash solution]

i: 2 j: 1 M1: 6 M2: 12

The 1st 2x2 matrix for the game «Crossroads»:

13 7

9 14

The 2st 2x2 matrix for the game «Crossroads»:

10 2

3 1

[The Pareto solution]:

i: 1 j: 1 M1: 13 M2: 10

i: 2 j: 2 M1: 14 M2: 1

[The Nash solution]

i: 1 j: 1 M1: 13 M2: 10

The 1st 2x2 matrix from the given option:

4 5

0 7

The 2st 2x2 matrix from the given option:

7 2

2 3

[The Pareto solution]

i: 1 j: 1 M1: 4 M2: 7

i: 2 j: 2 M1: 7 M2: 3

[The Nash solution]

i: 1 j: 1 M1: 4 M2: 7

i: 2 j: 2 M1: 7 M2: 3

[!] The solution is in mixed strategies

x: 0.167 0.833

y: 0.333 0.667

v1: 4.667

v2: 2.833

**Приложение Б**

исходный код **CMakeLists.txt**:

cmake\_minimum\_required(VERSION 2.8)

add\_executable(main

        main.cpp main.define.h

        matrix.h matrix.cpp

        vector.h vector.cpp

    )

исходный код **main.cpp**:

#include <iostream>

#include <iomanip>

#include <fstream>

#include <math.h>

#include <stdlib.h>

#include <time.h>

#include "main.define.h"

#include "vector.h"

#include "matrix.h"

namespace gameTheoryAndOperationsResearch {

    void \_gameTheoryAndOperationsResearch\_readfile() {

        std::string line;std::ifstream in(gameTheoryAndOperationsResearch\_filename);

        while (std::getline(in, line))

            std::cout << line << std::endl;

    }

    void \_gameTheoryAndOperationsResearch\_deletefile() {

        std::remove(gameTheoryAndOperationsResearch\_filename);

    }

    gameTheoryAndOperationsResearch\_matrix \_gameTheoryAndOperationsResearch\_matrix\_main;

    gameTheoryAndOperationsResearch\_vector \_gameTheoryAndOperationsResearch\_vector\_main;

    double M1[DEFINE\_N1][DEFINE\_N1] = { { 4, 5 },

                                        { 0, 7 } }; // Матрица для 1-го игрока

    double M2[DEFINE\_N1][DEFINE\_N1] = { { 7, 2 },

                                        { 2, 3} }; // Матрица для 2-го игрока

    // Проверка принадлежит ли решение с индексами i1 и j1 множеству Парето

    // Исходные матрицы передаются через указатели

    template<typename T>

    bool is\_lot\_of\_Pareto(T \*M1, T \*M2, int n, int i1, int j1)

    {

        T Val1 = \*(M1 + i1 \* n + j1);

        T Val2 = \*(M2 + i1 \* n + j1);

        for (int i = 0; i < n; i++)

            for (int j = 0; j < n; j++)

                if (!((i == i1) && (j == j1))) // Просматриваем все элементы кроме заданного

                    if ((\*(M1 + i \* n + j) >= Val1 && \*(M2 + i \* n + j) > Val2) ||

                        (\*(M1 + i \* n + j) > Val1 && \*(M2 + i \* n + j) >= Val2))

                        return false;  // Нашли решение, которое доминирует, решение не принадлежит множеству Парето

        return true; // Не нашли доминирующего решения, значит решение принадлежит множеству Парето

    }

    // Проверка принадлежит ли решение с индексами i1 и j1 множеству Неша

    template<typename T>

    bool is\_lot\_of\_Nesh(T \*M1, T \*M2, int n, int i1, int j1)

    {

        T Val1 = \*(M1 + i1 \* n + j1);

        T Val2 = \*(M2 + i1 \* n + j1);

        // Проверяем может ли первый игрок улучшить свое решение

        for (int i = 0; i < n; i++)

            if (i != i1)

                if (\*(M1 + i \* n + j1) > Val1)

                    return false; // Первый игрок может улучшить свое решение при заданном решении 2-го игрока, решение не принадлешит Нешу

        // Проверяем может ли второй игрок улучшить свое решение

        for (int j = 0; j < n; j++)

            if (j != j1)

                if (\*(M2 + i1 \* n + j) > Val2)

                    return false; // Второй игрок может улучшить свое решение при заданном решении 1-го игрока, решение не принадлехит Нешу

*if*(Val1 == Val2)

*return* *false*;

return true; // Решение принадлежит Нешу

    }

    // Печать матрицы c заголовком

    template<typename T>

    void  print\_matrix(std::ostream & out, char \*text, T \*M, int n)

    {

        out << text << ":" << std::endl; // Печать заголовка

        for (int i = 0; i < n; i++)

        {

            for (int j = 0; j < n; j++) {

                out.width(5);

                out << \*(M + i \* n + j);

            }

            out << std::endl;

        }

    }

    int \_main(int argc, char\* argv[])

    {

        srand(time(0));

        int M22\_1[DEFINE\_N2][DEFINE\_N2], M22\_2[DEFINE\_N2][DEFINE\_N2];   // Матрицы 10 х 10 для заполнения псевдослуч числами

        int N = 15;

        int M1\_1[DEFINE\_N1][DEFINE\_N1] = { { rand() % N, rand() % N },

                                           { rand() % N, rand() % N } }; // The 1st 2x2 matrix for the Prisoner's «Dilemma game»

        int M1\_2[DEFINE\_N1][DEFINE\_N1] = { { rand() % N, rand() % N },

                                           { rand() % N, rand() % N} }; // The 1st 2x2 matrix for the game  «Family Dispute»

        int M1\_3[DEFINE\_N1][DEFINE\_N1] = { { rand() % N, rand() % N },

                                           { rand() % N, rand() % N }}; // The 1st 2x2 matrix for the game "Crossroads"

        int M2\_1[DEFINE\_N1][DEFINE\_N1] = { { rand() % N, rand() % N },

                                           { rand() % N, rand() % N } }; // The 2st 2x2 matrix for the Prisoner's «Dilemma game»

        int M2\_2[DEFINE\_N1][DEFINE\_N1] = { { rand() % N, rand() % N },

                                           { rand() % N, rand() % N } }; // The 2st 2x2 matrix for the game  «Family Dispute»

        int M2\_3[DEFINE\_N1][DEFINE\_N1] = { { rand() % N, rand() % N },

                                           { rand() % N, rand() % N } }; // The 2st 2x2 matrix for the game «Crossroads»

        // Удаляем файл

        \_gameTheoryAndOperationsResearch\_deletefile();

        std::ofstream fout(gameTheoryAndOperationsResearch\_filename);

        fout << "\*\*\*\*GENERATION START\*\*\*\*" << std::endl;

        fout << "[Dilemma game]" << std::endl;

        print\_matrix(fout,  (char \*)"M1\_1", (int \*)M1\_1, (int)DEFINE\_N1);

        print\_matrix(fout,  (char \*)"M2\_1", (int \*)M2\_1, (int)DEFINE\_N1);

        fout << "[Family Dispute]" << std::endl;

        print\_matrix(fout,  (char \*)"M1\_2", (int \*)M1\_2, (int)DEFINE\_N1);

        print\_matrix(fout,  (char \*)"M2\_2", (int \*)M2\_2, (int)DEFINE\_N1);

        fout << "[Crossroads]" << std::endl;

        print\_matrix(fout,  (char \*)"M1\_3", (int \*)M1\_3, (int)DEFINE\_N1);

        print\_matrix(fout,  (char \*)"M2\_3", (int \*)M2\_3, (int)DEFINE\_N1);

        fout << "\*\*\*\*GENERATION END\*\*\*\*\n" << std::endl;

        // заполняем матрцы ПСЧ

        for(int i=0; i<DEFINE\_N2; i++) {

            for (int j = 0; j < DEFINE\_N2; j++)

            {

                M22\_1[i][j] = rand() % 20;

                M22\_2[i][j] = rand() % 20;

            }

        }

        print\_matrix(fout,  (char \*)"The 1st matrix is 10x10", (int \*)M22\_1, (int)DEFINE\_N2);

        print\_matrix(fout, (char \*)"The 2st matrix is 10x10", (int \*)M22\_2, DEFINE\_N2);

        // Поиск множества Парето

        fout << "\n[The Pareto solution]" << std::endl;

        for (int i = 0; i < DEFINE\_N2; i++)

            for (int j = 0; j < DEFINE\_N2; j++)

                if (is\_lot\_of\_Pareto((int \*)M22\_1, (int \*)M22\_2, DEFINE\_N2, i, j)) {// Нашли решение из множества Парето

                    fout << "\ti: " << (i + 1) << "\tj: " << (j + 1) << "\tM1: " << M22\_1[i][j] << "\tM2: " << M22\_2[i][j] << std::endl;

                }

        fout << "\n[The Nash solution]" << std::endl;

         bool flag = false;

        for (int i = 0; i < DEFINE\_N2; i++)

            for (int j = 0; j < DEFINE\_N2; j++)

                if (is\_lot\_of\_Nesh((int \*)M22\_1, (int \*)M22\_2, DEFINE\_N2, i, j)) // Нашли решение из множества Неша

                {

                    fout << "\ti: " << (i + 1)

                         << "\tj: " << (j + 1)

                         << "\tM1: " << M22\_1[i][j]

                         << "\tM2: " << M22\_2[i][j]

                         << std::endl;

                    flag = true;

                }

        if (!flag) {

                fout << "\n[!] There is no Nash solution in pure strategies" << std::endl;

        }

        print\_matrix(fout, (char \*)"The 1st 2x2 matrix for the Prisoner's «Dilemma game»", (int \*)M1\_1, DEFINE\_N1);

        print\_matrix(fout, (char \*)"The 2st 2x2 matrix for the Prisoner's «Dilemma game»", (int \*)M2\_1, DEFINE\_N1);

        // Поиск множества Парето

        fout << "\n[The Pareto solution]" << std::endl;

        for (int i = 0; i < DEFINE\_N1; i++)

            for (int j = 0; j < DEFINE\_N1; j++)

                if (is\_lot\_of\_Pareto((int \*)M1\_1, (int \*)M2\_1, DEFINE\_N1, i, j)) // Нашли решение из множества Парето

                {

                    fout << "\ti: " << (i + 1)

                         << "\tj: " << (j + 1)

                         << "\tM1: " << M1\_1[i][j]

                         << "\tM2: " << M2\_1[i][j]

                            << std::endl;

                }

        fout << "\n[The Nash solution]" << std::endl;

        flag = false;

        for (int i = 0; i < DEFINE\_N1; i++)

            for (int j = 0; j < DEFINE\_N1; j++)

                if (is\_lot\_of\_Nesh((int \*)M1\_1, (int \*)M2\_1, DEFINE\_N1, i, j)) // Нашли решение из множества Неша

                {

                    fout << "\ti: " << (i + 1) << "\tj: " << (j + 1) << "\tM1: " << M1\_1[i][j] << "\tM2: " << M2\_1[i][j] << std::endl;

                    flag = true;

                }

        if (!flag) {

            fout << "\n[!] There is no Nash solution in pure strategies" << std::endl;

        }

        print\_matrix(fout, (char \*)"The 1st 2x2 matrix for the game  «Family Dispute»", (int \*)M1\_2, DEFINE\_N1);

        print\_matrix(fout, (char \*)"The 2st 2x2 matrix for the game  «Family Dispute»", (int \*)M2\_2, DEFINE\_N1);

        // Поиск множества Парето

        fout << "\n[The Pareto solution]" << std::endl;

        for (int i = 0; i < DEFINE\_N1; i++)

            for (int j = 0; j < DEFINE\_N1; j++)

                if (is\_lot\_of\_Pareto((int \*)M1\_2, (int \*)M2\_2, DEFINE\_N1, i, j)) // Нашли решение из множества Парето

                {

                    fout << "\ti: " << (i + 1)

                         << "\tj: " << (j + 1)

                         << "\tM1: " << M1\_2[i][j]

                         << "\tM2: " << M2\_2[i][j]

                         << std::endl;

                }

        fout << "\n[The Nash solution]" << std::endl;

        flag = false;

        for (int i = 0; i < DEFINE\_N1; i++)

            for (int j = 0; j < DEFINE\_N1; j++)

                if (is\_lot\_of\_Nesh((int \*)M1\_2, (int \*)M2\_2, DEFINE\_N1, i, j)) // Нашли решение из множества Неша

                {

                    fout << "\ti: " << (i + 1)

                         << "\tj: " << (j + 1)

                         << "\tM1: " << M1\_2[i][j]

                         << "\tM2: " << M2\_2[i][j]

                         << std::endl;

                    flag = true;

                }

        if (!flag) {

                fout << "\n[!] There is no Nash solution in pure strategies" << std::endl;

        }

        print\_matrix(fout, (char \*)"The 1st 2x2 matrix for the game «Crossroads»", (int \*)M1\_3, DEFINE\_N1);

        print\_matrix(fout, (char \*)"The 2st 2x2 matrix for the game «Crossroads»", (int \*)M2\_3, DEFINE\_N1);

        // Поиск множества Парето

        fout << "\n[The Pareto solution]:" << std::endl;

        for (int i = 0; i < DEFINE\_N1; i++)

            for (int j = 0; j < DEFINE\_N1; j++)

                if (is\_lot\_of\_Pareto((int \*)M1\_3, (int \*)M2\_3, DEFINE\_N1, i, j)) // Нашли решение из множества Парето

                {

                    fout << "\ti: " << (i + 1)

                         << "\tj: " << (j + 1)

                         << "\tM1: " << M1\_3[i][j]

                         << "\tM2: " << M2\_3[i][j]

                         << std::endl;

                }

        fout << "\n[The Nash solution]" << std::endl;

        flag = false;

        for (int i = 0; i < DEFINE\_N1; i++)

            for (int j = 0; j < DEFINE\_N1; j++)

                if (is\_lot\_of\_Nesh((int \*)M1\_3, (int \*)M2\_3, DEFINE\_N1, i, j)) // Нашли решение из множества Неша

                {

                    fout << "\ti: " << (i + 1)

                         << "\tj: " << (j + 1)

                         << "\tM1: " << M1\_3[i][j]

                         << "\tM2: " << M2\_3[i][j]

                         << std::endl;

                    flag = true;

                }

        if (!flag) {

                fout << "\n[!] There is no Nash solution in pure strategies" << std::endl;

        }

        print\_matrix(fout, (char \*)"The 1st 2x2 matrix from the given option", (double \*)M1, DEFINE\_N1);

        print\_matrix(fout, (char \*)"The 2st 2x2 matrix from the given option", (double \*)M2, DEFINE\_N1);

        // Поиск множества Парето

        fout << "\n[The Pareto solution]" << std::endl;

        for (int i = 0; i < DEFINE\_N1; i++)

            for (int j = 0; j < DEFINE\_N1; j++)

                if (is\_lot\_of\_Pareto((double \*)M1, (double \*)M2, DEFINE\_N1, i, j)) // Нашли решение из множества Парето

                {

                    fout << "\ti: " << (i + 1)

                         << "\tj: " << (j + 1)

                         << "\tM1: " << M1[i][j]

                         << "\tM2: " << M2[i][j]

                         << std::endl;

                }

        fout << "\n[The Nash solution]" << std::endl;

        flag = false;

        int Num=0;

        for (int i = 0; i < DEFINE\_N1; i++)

            for (int j = 0; j < DEFINE\_N1; j++)

                if (is\_lot\_of\_Nesh((double \*)M1, (double \*)M2, DEFINE\_N1, i, j)) // Нашли решение из множества Неша

                {

                    fout << "\ti: " << (i + 1)

                         << "\tj: " << (j + 1)

                         << "\tM1: " << M1[i][j]

                         << "\tM2: " << M2[i][j]

                         << std::endl;

                    flag = true;

                    Num++;

                }

        if (!flag) {

                fout << "\n[!] There is no Nash solution in pure strategies" << std::endl;

        }

        fout<<"\n[!] The solution is in mixed strategies"<<std::endl;

        if (Num!=1)

        {

            double \*p = new double[DEFINE\_N1]; double \*q = new double[DEFINE\_N1];

            double \*\*c = new double \*[DEFINE\_N1]; // Обратная матрица

            for (int i = 0; i < DEFINE\_N1; i++)

                c[i] = new double[DEFINE\_N1];

            \_gameTheoryAndOperationsResearch\_matrix\_main.get\_inverse\_matrix(M2, c, DEFINE\_N1); // Получаем обратную матрицу

            double \* u = new double[DEFINE\_N1]; // Вспомогательный вектор из всех 1

            for (int i = 0; i < DEFINE\_N1; i++)

                u[i] = 1;

            double \* r1 = new double[DEFINE\_N1]; // Вектор для промежуточных вычислений

            \_gameTheoryAndOperationsResearch\_vector\_main.composition\_vector\_matrix(u, c, r1, DEFINE\_N1, DEFINE\_N1);

            double v2 = \_gameTheoryAndOperationsResearch\_vector\_main.composition\_vector\_vector(r1, u, DEFINE\_N1);

            \_gameTheoryAndOperationsResearch\_vector\_main.composition\_vector\_matrix(u, c, p, DEFINE\_N1, DEFINE\_N1);

            for (int i = 0; i < DEFINE\_N1; i++)

                p[i] /= v2;

            \_gameTheoryAndOperationsResearch\_vector\_main.print\_vector(fout, (char \*)"x", p, DEFINE\_N1);

            \_gameTheoryAndOperationsResearch\_matrix\_main.get\_inverse\_matrix(M1, c, DEFINE\_N1); // Получаем обратную матрицу

            \_gameTheoryAndOperationsResearch\_vector\_main.composition\_vector\_matrix(u, c, r1, DEFINE\_N1, DEFINE\_N1);

            double v1 = \_gameTheoryAndOperationsResearch\_vector\_main.composition\_vector\_vector(r1, u, DEFINE\_N1);

            \_gameTheoryAndOperationsResearch\_vector\_main.composition\_matrix\_vector(c, u, q, DEFINE\_N1, DEFINE\_N1);

            for (int i = 0; i < DEFINE\_N1; i++)

                q[i] /= v1;

            \_gameTheoryAndOperationsResearch\_vector\_main.print\_vector(fout, (char \*)"y", q, DEFINE\_N1);

            fout << std::fixed << std::setprecision(3);

            fout << "v1:    " << 1/v1 << std::endl;

            fout << "v2:    " << 1/v2 << std::endl;

        }

        else

        {

            fout<<"\n[!] One player has a strictly dominant strategy, there is only one Nash solution"<<std::endl;

        }

        // Считываем файл

        \_gameTheoryAndOperationsResearch\_readfile();

        return 1;

        return 0;

    }

}

int main(int argc, char\* argv[])

{

    gameTheoryAndOperationsResearch::\_main(argc, argv);

    return 0;

}

исходный код **main.define.h**:

#ifndef MAIN\_DEFINE\_H

#define MAIN\_DEFINE\_H

#define DEFINE\_N1  2

#define DEFINE\_N2 10

#define gameTheoryAndOperationsResearch\_filename "gameTheoryAndOperationsResearch\_l4.txt"

#endif // MAIN\_DEFINE\_H

исходный код **matrix.cpp**:

#include "matrix.h"

namespace gameTheoryAndOperationsResearch {

    // Далее функции для аналитического метода обратной матрицы

    double gameTheoryAndOperationsResearch\_matrix::get\_determinant\_matrix(double \*\*p, int n) // Вычисление определителя матрицы

    {

        if (n == 1) {

            return p[0][0];

        }

        if (n == 2) {

            return p[0][0] \* p[1][1] - p[0][1] \* p[1][0];

        }

        else

        {

            double \*\*p2 = new double \*[n - 1];

            for (int i = 0; i < n - 1; i++)

                p2[i] = new double[n - 1];

            int zn = 1;

            double Det = 0;

            for (int k = 0; k < n; k++, zn \*= -1)

            {

                if (p[0][k] == 0)

                    continue;

                for (int i = 1; i < n; i++) {

                    for (int j = 0; j < n; j++)

                    {

                        if (j < k) p2[i - 1][j] = p[i][j];

                        if (j > k) p2[i - 1][j - 1] = p[i][j];

                    }

                }

                Det += zn \* p[0][k] \* get\_determinant\_matrix(p2, n - 1);

            }

            for (int i = 0; i < n - 1; i++)

                delete[] p2[i];

            delete[] p2;

            return Det;

        }

    }

    void gameTheoryAndOperationsResearch\_matrix::get\_inverse\_matrix(double p[DEFINE\_N1][DEFINE\_N1], double \*\*pObr, int n) // Вычисление обратной матрицы для матрицы

    {

        double Det = 0;

        if (n == 2)

        {

            Det = p[0][0] \* p[1][1] - p[0][1] \* p[1][0];

            pObr[0][0] = p[1][1];

            pObr[0][1] = p[1][0] \* -1;

            pObr[1][0] = p[0][1] \* -1;

            pObr[1][1] = p[0][0];

        }

        else

        {

            double \*\*p2 = new double \*[n - 1];

            for (int i = 0; i < n - 1; i++)

                p2[i] = new double[n - 1];

            int zn = 1;

            Det = 0;

            for (int k0 = 0; k0 < n; k0++)

            {

                for (int k = 0; k < n; k++)

                {

                    if ((k + k0) % 2)

                        zn = -1;

                    else

                        zn = 1;

                    //  if (p[0][k] == 0) continue;

                    for (int i = 0; i < n; i++) {

                        for (int j = 0; j < n; j++)

                        {

                            if (i > k0)

                            {

                                if (j < k) p2[i - 1][j] = p[i][j];

                                if (j > k) p2[i - 1][j - 1] = p[i][j];

                            }

                            if (i < k0)

                            {

                                if (j < k) p2[i][j] = p[i][j];

                                if (j > k) p2[i][j - 1] = p[i][j];

                            }

                        }

                    }

                    double opr = get\_determinant\_matrix(p2, n - 1);

                    pObr[k0][k] = zn \* opr;

                    if (k0 == 0)

                        Det += p[k0][k] \* pObr[k0][k];

                }

            }

            for (int i = 0; i < n - 1; i++)

                delete[] p2[i];

            delete[] p2;

        }

        // Транспонируем матрицу

        for (int i = 0; i < n - 1; i++) for (int j = i + 1; j < n; j++)

        {

            double buf = pObr[i][j]; pObr[i][j] = pObr[j][i];

            pObr[j][i] = buf;

        }

        for (int i = 0; i < n; i++) for (int j = 0; j < n; j++) {

            pObr[i][j] /= Det;

            if (fabs(pObr[i][j]) < 1e-9) pObr[i][j] = 0;

        }

    }

}

исходный код **matrix.h**:

#ifndef MATRIX\_H

#define MATRIX\_H

#include "main.define.h"

#include <iostream>

#include <iomanip>

#include <fstream>

#include <math.h>

#include <stdlib.h>

namespace gameTheoryAndOperationsResearch {

    class gameTheoryAndOperationsResearch\_matrix

    {

        public:

            double get\_determinant\_matrix(double\*\*, int);

            void get\_inverse\_matrix(double p[DEFINE\_N1][DEFINE\_N1], double\*\*, int);

        };

}

#endif // MATRIX\_H

исходный код **vector.cpp**:

#include "vector.h"

namespace gameTheoryAndOperationsResearch {

    // Произведение вектора на матрицу

    void gameTheoryAndOperationsResearch\_vector::composition\_vector\_matrix(double \*v, double \*\*matr, double\* v\_out, int n, int m)

    {

        for (int i = 0; i < m; i++)

        {

            v\_out[i] = 0;

            for (int j = 0; j < n; j++)

                v\_out[i] += v[j] \* matr[j][i];

        }

    }

    // Произведение матрицы на вектор

    void gameTheoryAndOperationsResearch\_vector::composition\_matrix\_vector(double \*\*matr, double \*v,  double\* v\_out, int n, int m)

    {

        for (int i = 0; i < n; i++)

        {

            v\_out[i] = 0;

            for (int j = 0; j < m; j++)

                v\_out[i] += v[j] \* matr[i][j];

        }

    }

    // Произведение вектор на вектор - на выходе число (скалярное произведение)

    double gameTheoryAndOperationsResearch\_vector::composition\_vector\_vector(double \*v1, double \*v2, int n)

    {

        double rez = 0;

        for (int i = 0; i < n; i++)

            rez += v1[i] \* v2[i];

        return rez;

    }

    void gameTheoryAndOperationsResearch\_vector::print\_vector(std::ostream& out, char \* str, double \*p, int n)

    {

        out.precision(3);

        out << str << ": ";

        for (int i = 0; i < n; i++)

            out << p[i] << " ";

        out << std::endl;

    }

}

исходный код **vector.h**:

#ifndef VECTOR\_H

#define VECTOR\_H

#include <iostream>

#include <iomanip>

#include <fstream>

#include <math.h>

#include <stdlib.h>

namespace gameTheoryAndOperationsResearch {

    class gameTheoryAndOperationsResearch\_vector

    {

    public:

        void composition\_vector\_matrix(double \*, double \*\*, double\*, int, int);

        void composition\_matrix\_vector(double \*\*, double \*,  double\*, int, int);

        double composition\_vector\_vector(double \*, double \*, int);

        void print\_vector(std::ostream& , char \*, double \*, int );

    };

}

#endif // VECTOR\_H

**Приложение В**

**Тест программы №1:**

\*\*\*\*GENERATION START\*\*\*\*

[Dilemma game]

M1\_1:

7 12

0 13

M2\_1:

8 6

12 3

[Family Dispute]

M1\_2:

9 0

14 2

M2\_2:

2 4

6 1

[Crossroads]

M1\_3:

14 4

7 5

M2\_3:

12 12

8 5

\*\*\*\*GENERATION END\*\*\*\*

The 1st matrix is 10x10:

16 0 11 9 0 11 13 0 9 15

2 1 12 10 16 3 10 7 1 19

15 2 7 13 1 16 7 1 18 16

19 7 18 19 2 7 10 13 8 2

0 13 3 7 17 14 7 15 5 4

15 14 10 15 10 0 10 12 19 16

16 14 5 16 11 19 0 19 15 4

14 1 6 1 15 15 1 6 9 5

16 7 17 10 1 13 12 15 17 10

10 5 18 5 11 2 19 12 17 8

The 2st matrix is 10x10:

16 15 13 19 17 1 2 2 8 12

12 2 15 2 19 8 16 4 6 17

18 12 2 17 1 11 19 18 9 4

16 5 4 4 6 17 17 0 5 2

17 19 15 19 10 13 15 13 3 17

0 18 13 11 19 2 9 14 16 6

4 12 9 16 10 13 18 4 15 4

7 14 1 18 1 13 7 13 2 13

0 6 12 11 18 11 6 4 4 10

19 5 12 8 17 2 9 10 3 7

[The Pareto solution]

i: 2 j: 5 M1: 16 M2: 19

i: 2 j: 10 M1: 19 M2: 17

[The Nash solution]

[!] There is no Nash solution in pure strategies

The 1st 2x2 matrix for the Prisoner's «Dilemma game»:

7 12

0 13

The 2st 2x2 matrix for the Prisoner's «Dilemma game»:

8 6

12 3

[The Pareto solution]

i: 1 j: 1 M1: 7 M2: 8

i: 1 j: 2 M1: 12 M2: 6

i: 2 j: 1 M1: 0 M2: 12

i: 2 j: 2 M1: 13 M2: 3

[The Nash solution]

i: 1 j: 1 M1: 7 M2: 8

The 1st 2x2 matrix for the game «Family Dispute»:

9 0

14 2

The 2st 2x2 matrix for the game «Family Dispute»:

2 4

6 1

[The Pareto solution]

i: 2 j: 1 M1: 14 M2: 6

[The Nash solution]

i: 2 j: 1 M1: 14 M2: 6

The 1st 2x2 matrix for the game «Crossroads»:

14 4

7 5

The 2st 2x2 matrix for the game «Crossroads»:

12 12

8 5

[The Pareto solution]:

i: 1 j: 1 M1: 14 M2: 12

[The Nash solution]

i: 1 j: 1 M1: 14 M2: 12

The 1st 2x2 matrix from the given option:

4 5

0 7

The 2st 2x2 matrix from the given option:

7 2

2 3

[The Pareto solution]

i: 1 j: 1 M1: 4 M2: 7

i: 2 j: 2 M1: 7 M2: 3

[The Nash solution]

i: 1 j: 1 M1: 4 M2: 7

i: 2 j: 2 M1: 7 M2: 3

[!] The solution is in mixed strategies

x: 0.167 0.833

y: 0.333 0.667

v1: 4.667

v2: 2.833

**Тест программы №2:**

**\*\*\*\*GENERATION START\*\*\*\***

**[Dilemma game]**

**M1\_1:**

**6 0**

**7 9**

**M2\_1:**

**3 2**

**9 12**

**[Family Dispute]**

**M1\_2:**

**14 6**

**11 11**

**M2\_2:**

**6 9**

**3 7**

**[Crossroads]**

**M1\_3:**

**13 1**

**0 5**

**M2\_3:**

**5 14**

**4 12**

**\*\*\*\*GENERATION END\*\*\*\***

**The 1st matrix is 10x10:**

**9 8 8 10 15 9 16 19 9 11**

**15 16 1 12 10 3 13 1 1 10**

**9 8 17 6 6 3 7 17 8 11**

**15 5 17 14 14 6 16 9 5 8**

**3 11 15 6 13 16 7 9 5 15**

**12 15 5 9 19 13 7 15 8 5**

**16 3 6 4 8 4 2 9 11 18**

**8 8 3 19 6 17 0 9 15 19**

**14 9 17 1 11 13 7 15 14 3**

**12 2 9 12 19 12 1 14 17 17**

**The 2st matrix is 10x10:**

**8 1 13 17 16 17 12 11 6 14**

**12 10 15 14 2 16 17 5 6 14**

**2 2 15 5 13 8 3 14 3 3**

**5 7 7 1 5 0 3 1 3 2**

**1 19 0 12 19 10 11 13 7 4**

**2 10 12 14 11 6 11 6 11 14**

**18 18 14 5 8 8 9 2 12 4**

**15 5 10 4 5 12 9 5 18 16**

**15 15 19 10 19 1 8 11 14 14**

**16 12 2 9 9 18 11 6 3 19**

**[The Pareto solution]**

**i: 8 j: 10 M1: 19 M2: 16**

**i: 9 j: 3 M1: 17 M2: 19**

**i: 10 j: 10 M1: 17 M2: 19**

**[The Nash solution]**

**i: 3 j: 3 M1: 17 M2: 15**

**i: 4 j: 3 M1: 17 M2: 7**

**i: 7 j: 1 M1: 16 M2: 18**

**i: 9 j: 3 M1: 17 M2: 19**

**The 1st 2x2 matrix for the Prisoner's «Dilemma game»:**

**6 0**

**7 9**

**The 2st 2x2 matrix for the Prisoner's «Dilemma game»:**

**3 2**

**9 12**

**[The Pareto solution]**

**i: 2 j: 2 M1: 9 M2: 12**

**[The Nash solution]**

**i: 2 j: 2 M1: 9 M2: 12**

**The 1st 2x2 matrix for the game «Family Dispute»:**

**14 6**

**11 11**

**The 2st 2x2 matrix for the game «Family Dispute»:**

**6 9**

**3 7**

**[The Pareto solution]**

**i: 1 j: 1 M1: 14 M2: 6**

**i: 1 j: 2 M1: 6 M2: 9**

**i: 2 j: 2 M1: 11 M2: 7**

**[The Nash solution]**

**i: 2 j: 2 M1: 11 M2: 7**

**The 1st 2x2 matrix for the game «Crossroads»:**

**13 1**

**0 5**

**The 2st 2x2 matrix for the game «Crossroads»:**

**5 14**

**4 12**

**[The Pareto solution]:**

**i: 1 j: 1 M1: 13 M2: 5**

**i: 1 j: 2 M1: 1 M2: 14**

**i: 2 j: 2 M1: 5 M2: 12**

**[The Nash solution]**

**i: 2 j: 2 M1: 5 M2: 12**

**The 1st 2x2 matrix from the given option:**

**4 5**

**0 7**

**The 2st 2x2 matrix from the given option:**

**7 2**

**2 3**

**[The Pareto solution]**

**i: 1 j: 1 M1: 4 M2: 7**

**i: 2 j: 2 M1: 7 M2: 3**

**[The Nash solution]**

**i: 1 j: 1 M1: 4 M2: 7**

**i: 2 j: 2 M1: 7 M2: 3**

**[!] The solution is in mixed strategies**

**x: 0.167 0.833**

**y: 0.333 0.667**

**v1: 4.667**

**v2: 2.833**

**Тест программы №3:**

**\*\*\*\*GENERATION START\*\*\*\***

**[Dilemma game]**

**M1\_1:**

**6 0**

**6 5**

**M2\_1:**

**6 0**

**1 0**

**[Family Dispute]**

**M1\_2:**

**9 2**

**4 13**

**M2\_2:**

**14 5**

**11 0**

**[Crossroads]**

**M1\_3:**

**4 6**

**4 3**

**M2\_3:**

**10 7**

**8 10**

**\*\*\*\*GENERATION END\*\*\*\***

**The 1st matrix is 10x10:**

**2 15 2 2 15 12 12 17 17 7**

**9 3 4 4 2 12 6 19 18 17**

**3 4 12 14 14 7 2 10 6 16**

**8 6 5 1 2 19 9 7 9 3**

**17 9 2 0 0 12 16 13 7 0**

**1 18 2 1 13 1 12 15 15 8**

**0 15 5 15 0 8 12 12 5 3**

**6 6 17 19 7 10 12 6 9 6**

**12 9 3 9 1 16 6 12 17 8**

**11 14 17 12 14 10 6 0 18 19**

**The 2st matrix is 10x10:**

**2 3 6 7 3 7 6 8 4 11**

**14 19 12 1 10 16 17 14 15 7**

**19 8 3 18 14 16 16 18 0 8**

**17 15 8 14 9 7 16 7 2 19**

**16 1 15 10 17 9 15 12 19 2**

**11 9 19 16 12 11 5 9 11 15**

**19 0 6 17 5 5 18 19 3 19**

**8 6 2 14 15 13 11 13 5 0**

**18 17 17 11 5 4 19 4 7 11**

**19 17 5 0 18 7 8 9 17 16**

**[The Pareto solution]**

**i: 3 j: 4 M1: 14 M2: 18**

**i: 7 j: 8 M1: 12 M2: 19**

**i: 10 j: 5 M1: 14 M2: 18**

**i: 10 j: 9 M1: 18 M2: 17**

**i: 10 j: 10 M1: 19 M2: 16**

**[The Nash solution]**

**[!] There is no Nash solution in pure strategies**

**The 1st 2x2 matrix for the Prisoner's «Dilemma game»:**

**6 0**

**6 5**

**The 2st 2x2 matrix for the Prisoner's «Dilemma game»:**

**6 0**

**1 0**

**[The Pareto solution]**

**[The Nash solution]**

**i: 2 j: 1 M1: 6 M2: 1**

**The 1st 2x2 matrix for the game «Family Dispute»:**

**9 2**

**4 13**

**The 2st 2x2 matrix for the game «Family Dispute»:**

**14 5**

**11 0**

**[The Pareto solution]**

**i: 1 j: 1 M1: 9 M2: 14**

**i: 2 j: 2 M1: 13 M2: 0**

**[The Nash solution]**

**i: 1 j: 1 M1: 9 M2: 14**

**The 1st 2x2 matrix for the game «Crossroads»:**

**4 6**

**4 3**

**The 2st 2x2 matrix for the game «Crossroads»:**

**10 7**

**8 10**

**[The Pareto solution]:**

**i: 1 j: 1 M1: 4 M2: 10**

**i: 1 j: 2 M1: 6 M2: 7**

**[The Nash solution]**

**i: 1 j: 1 M1: 4 M2: 10**

**The 1st 2x2 matrix from the given option:**

**4 5**

**0 7**

**The 2st 2x2 matrix from the given option:**

**7 2**

**2 3**

**[The Pareto solution]**

**i: 1 j: 1 M1: 4 M2: 7**

**i: 2 j: 2 M1: 7 M2: 3**

**[The Nash solution]**

**i: 1 j: 1 M1: 4 M2: 7**

**i: 2 j: 2 M1: 7 M2: 3**

**[!] The solution is in mixed strategies**

**x: 0.167 0.833**

**y: 0.333 0.667**

**v1: 4.667**

**v2: 2.833**