МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«КУЗБАССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Т.Ф. ГОРБАЧЕВА»

Кафедра информационных и автоматизированных систем

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

по профессиональному модулю

ПМ 06 «Сопровождение информационных систем»

Выполнил:
студент группы ИСт-202
Алексеева Т. Н.

Руководитель практики:
преподаватель
Сыркин И. С.

Оценка_____

«___»_____20___ г.

_____ *Сыркин И. С.*

Подпись (расшифровка подписи)

Кемерово 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Программа учебной практики является частью программы подготовки специалистов среднего звена в соответствии с ФГОС по специальности СПО 09.02.07 «Информационные системы и программирование» в части освоения основного вида профессиональной деятельности (ВПД). Данная практика направлена на формирование следующих компетенций:

ОК 01 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам

ОК 02 Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности

ОК 03 Планировать и реализовывать собственное профессиональное и личностное развитие

ОК 04 Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами

ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста

ОК 06 Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, применять стандарты антикоррупционного поведения

ОК 07 Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях

ОК 08 Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности

ОК 09 Использовать информационные технологии в профессиональной деятельности

ОК 10 Пользоваться профессиональной документацией на государственном и иностранном языках

ПК 6.1 Разрабатывать техническое задание на сопровождение информационной системы

ПК 6.2 Выполнять исправление ошибок в программном коде информационной системы

ПК 6.3 Разрабатывать обучающую документацию для пользователей информационной системы

ПК 6.4 Оценивать качество и надежность функционирования информационной системы в соответствии с критериями технического задания

ПК 6.5 Осуществлять техническое сопровождение, обновление и восстановление данных информационной системы в соответствии с техническим заданием

1 ВИДЫ ИНФОРМАЦИОННЫХ СИСТЕМ

1.1 Состав и классификация информационных систем

1.1.1 Общие сведения

1.1.2 Индивидуальное задание

1.1.2.1 Задание 1

Условие задачи представлено на рисунке 1.1.

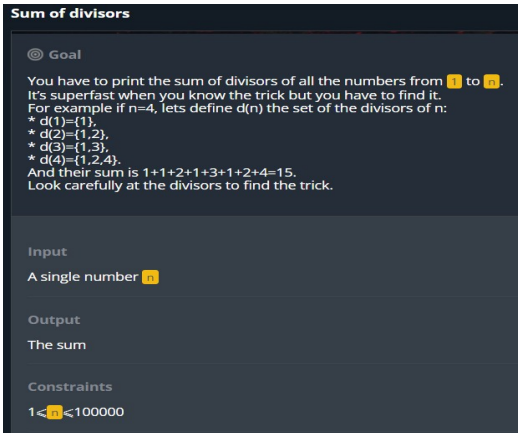


Рисунок 1.1

Sum of divisors

```
using System;
using System.Linq;
using System.IO;
using System.Text;
using System.Collections;
using System.Collections.Generic;

public class Solution
    {
        public static void Main(string[] args)
        {
            long n = long.Parse(Console.ReadLine());

            long sum = 0;
            for (long i = n - 1; i >= 0; i--)
            {
                sum += GetDivisors(n--).Sum(); // Sum позволяют сложить все элементы
списка
            }

            Console.WriteLine(sum);
        }

        public static IEnumerable<long> GetDivisors(long n)
```

```
        {
            var result = new List<long>();

            for (long i = 1; i <= Math.Sqrt(n) + 1; i++)
            {
                if (n % i == 0)
                {
                    // Если слагаемые равны, добавляем один элемент
                    if (n / i == i)
                    {
                        result.Add(i);
                    }
                    else // Иначе выводим оба
                    {
                        result.Add(i);
                        result.Add(n / i);
                    }
                }
            }
            return result.Distinct();
        }
    }
```

1.1.2.2 Задание 2

Условие задачи представлено на рисунке 1.2.

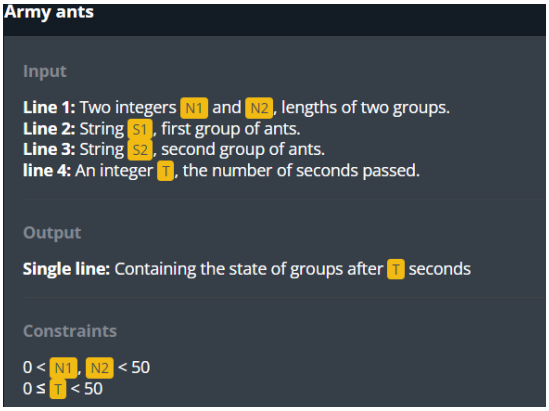


Рисунок 1.2

Army ants

```
using System;
using System.Linq;
using System.IO;
using System.Text;
using System.Collections;
using System.Collections.Generic;

class Solution
{
    static void Main(string[] args)
    {
        string[] inputs = Console.ReadLine().Split(' ');
        int N1 = int.Parse(inputs[0]); // Длина первой группы муравьёв
```

```
        int N2 = int.Parse(inputs[1]); // Длина второй группы муравьёв
        string S1 = Console.ReadLine(); // Первая группа муравьёв
        string S2 = Console.ReadLine(); // Вторая группа муравьёв
        int T = int.Parse(Console.ReadLine()); // Время прохождения муравь-ёв через
друг друга
        S1 = string.Join("",S1.Reverse());
        char[] result = (S1+S2).ToArray(); // Преобразование переменной в массив, для
работы с индексом элементов

        for(int i = 1; i <= T; i++)
        {
            for(int j = 0; j < result.Length- 1 ; j++)
            {
                char f = result[j]; // Первый элемент списка
                char n = result[j+1]; // Следующий элемент списка

                if(S1.Contains(f) && S2.Contains(n)) // Если первая группа со-держит
первый элемент списка и вторая группа содержит второй элемент списка
                {
                    var temp = result[j]; // Temp - переменная служащая ячейкой памяти
для запоминания сортировочных элементов
                    result[j] = result[j+1];
                    result[j+1] = temp;
                    j++;
                }
            }
        }
        Console.WriteLine(string.Join("", result)); // Используем метод Join, для
вывода сцеплённых букв.
}
}
```

1.2 Классификация информационных систем

1.2.1 Общие сведения

1.2.2 Индивидуальное задание

1.2.2.1 Задание 1

Условие задачи представлено на рисунке 1.3.

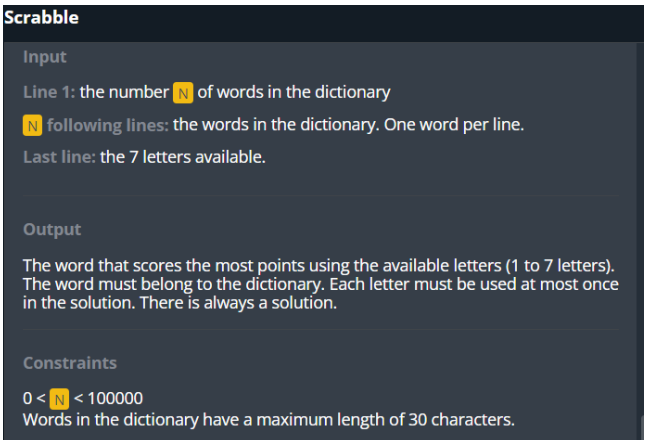


Рисунок 1.3

Scrabble

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text.RegularExpressions;

    public class Solution
    {
        private static IDictionary<IEnumerable<char>, int> scrabbleRate = new Dictio-
nary<IEnumerable<char>, int>
        {
            { new List<char>  { 'e', 'a', 'i', 'o', 'n', 'r', 't', 'l', 's', 'u' }, 1
},
            { new List<char>  { 'd', 'g' },                                          2
},
            { new List<char>  { 'b', 'c', 'm', 'p' },                                3
},
            { new List<char>  { 'f', 'h', 'v', 'w', 'y' },                           4
},
            { new List<char>  { 'k' },                                               5
},
            { new List<char>  { 'j', 'x' },                                          8
},
            { new List<char>  { 'q', 'z' },                                          10
}
        };

        public static bool IsPossibleWord(IEnumerable<char> letters, string word)
        {
            while (true)
            {
                if (!letters.Any())
                    if (word.Length > 0)
                        return false;
                    else
                        return true;
                var character = letters.Take(1).First();
                letters = string.Concat(letters.Skip(1));
                if (!word.Contains(character))
                    continue;
                else
                {
                    var regex = new Regex(Regex.Escape(character.ToString()));
                    word = regex.Replace(word, string.Empty, 1);
                }
            }
        }

        public static int LetterRate(char character)
        {
            return scrabbleRate.Where(listCharacter =>
listCharacter.Key.Contains(character)).Select(rate => rate.Value).First();
        }
```

```
        public static void Main(string[] args)
        {
            int N = int.Parse(Console.ReadLine());

            var dictionnaryWords = Enumerable.Range(0, N).Select(x => Console.Read-
Line()).ToList();

            var letters = Console.ReadLine();

            var possibleWord = dictionnaryWords.Where(word => word.Length <= letter-
s.Length)
                                               .Where(word => word.All(character =>
letters.Contains(character)))
                                               .Where(word => IsPossibleWord(letters,
word)).ToList();

            var bestWordRated = possibleWord.OrderByDescending(x =>
WordRate(x)).First();

            Console.WriteLine(bestWordRated);
        }

        public static int WordRate(string word)
        {
            return word.Sum(LetterRate);
        }
    }
```

1.2.2.2 Задание 2

Условие задачи представлено на рисунке 1.4.

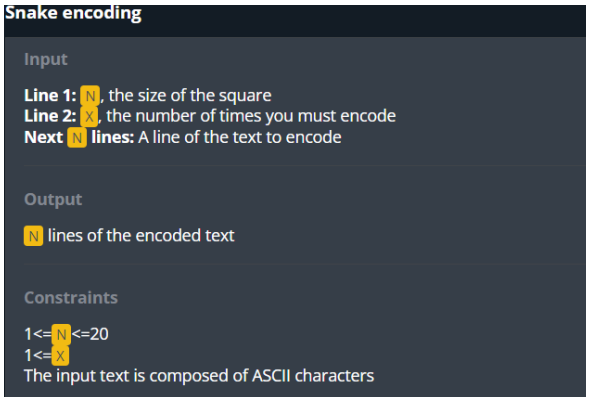


Рисунок 1.4

Snake encoding

```
using System;

internal class Solution
{
    private static void Encode(string[,] arr)
    {
        var N = arr.GetLength(0);
        var j = 0;
        var last = N % 2 == 0 ? arr[N - 1, N - 1] : arr[0, N - 1];
```

```
            while (j < N)
            {
                if (j % 2 == 0)
                {
                    for (int i = N - 1; i >= 0; i--)
                    {
                        var temp = arr[i, j];
                        arr[i, j] = last;
                        last = temp;
                    }
                }
                else
                {
                    for (int i = 0; i < N; i++)
                    {
                        var temp = arr[i, j];
                        arr[i, j] = last;
                        last = temp;
                    }
                }
                j++;
            }
        }

        private static void Main(string[] args)
        {
            int N = int.Parse(Console.ReadLine());
            int X = int.Parse(Console.ReadLine());
            var arr = new string[N, N];
            int j;
            for (int i = 0; i < N; i++)
            {
                var LINE = Console.ReadLine().ToCharArray();
                for (j = 0; j < N; j++)
                {
                    arr[i, j] = LINE[j].ToString();
                }
            }

            for (int i = 0; i < X; i++)
            {
                Encode(arr);
            }

            for (int i = 0; i < N; i++)
            {
                for (j = 0; j < N; j++)
                {
                    Console.Write(arr[i, j]);
                }
                Console.WriteLine();
            }
        }
    }
```

2 СОПРОВОЖДЕНИЕ УЧЁТНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

2.1 Архитектура учетной информационной системы

2.1.1 Общие сведения

2.1.2 Индивидуальное задание

2.1.2.1 Задание 1

Условие задачи представлено на рисунке 2.1.

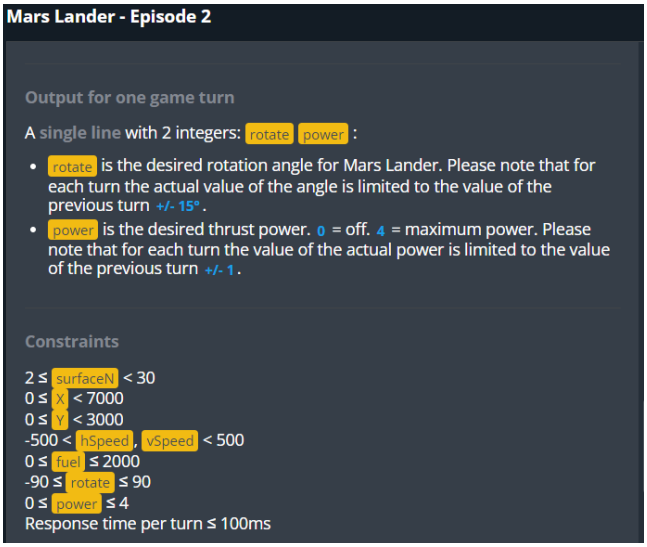


Рисунок 2.1

Mars Lander – Episode 2

```
using System;
using System.Linq;
using System.IO;
using System.Text;
using System.Collections;
using System.Collections.Generic;

/**
 * Auto-generated code below aims at helping you parse
 * the standard input according to the problem statement.
 **/
class Player
{
    static void Main(string[] args)
    {
        string[] inputs;
        int surfaceN = int.Parse(Console.ReadLine()); // the number of points used to
draw the surface of Mars.
        int landingPointX1 = 0;
        int landingPointX2 = 0;
        int landingPointMiddle = 0;
```

```
        int landingPointY = 0;
        int done = 0;

        for (int i = 0; i < surfaceN; i++)
        {
            inputs = Console.ReadLine().Split(' ');
            int landX = int.Parse(inputs[0]); // X coordinate of a surface point. (0
to 6999)
            int landY = int.Parse(inputs[1]); // Y coordinate of a surface point. By
linking all the points together in a sequential fashion, you form the surface of Mars.
            Console.Error.WriteLine($"landx = {landX}; landY = {landY}");

            if (landY == landingPointY && done == 0)
            {
                landingPointY = landY;
                landingPointX2 = landX;
                done = 1;
            } else if (landY != landingPointY && done == 0)
            {
                landingPointY = landY;
                landingPointX1 = landX;
            }

            landingPointMiddle = (landingPointX1 + landingPointX2) / 2 ;

            Console.Error.WriteLine($"H={landingPointY};X1={landingPointX1};X2={land-
ingPointX2};XM={landingPointMiddle}");
        }

        // safety and physic engine:
        double g = 3.711, maxMthrust = 4;
        int maxHspeed = 20, maxVspeed = 40;

        //different Variables for math stuff
        double mSpeed = 0.00;
        double desiredAngle = ConvertRadiansToDegrees(Math.Acos(g / maxMthrust));

        int mRotate = 0, mThrust = 4;
        int mDistanceToLandingArea = 0;

        // game loop
        while (true)
        {
            inputs = Console.ReadLine().Split(' ');
            int xMars = int.Parse(inputs[0]);
            int yMars = int.Parse(inputs[1]);
            int hSpeed = int.Parse(inputs[2]); // the horizontal speed (in m/s), can
be negative.
            int vSpeed = int.Parse(inputs[3]); // the vertical speed (in m/s), can be
negative.
            int fuel = int.Parse(inputs[4]); // the quantity of remaining fuel in
liters.
            int rotate = int.Parse(inputs[5]); // the rotation angle in degrees (-90
to 90).
            int power = int.Parse(inputs[6]); // the thrust power (0 to 4).
```

```
            mSpeed = ConvertRadiansToDegrees(Math.Sqrt(Math.Pow(hSpeed, 2) +
Math.Pow(vSpeed, 2)));

            // if not landing area
            if (!((landingPointX1 < xMars) && (xMars < landingPointX2)))
            {
                Console.Error.WriteLine($"not landing Area");
                // Check for direction
                if ((xMars < landingPointX1 && hSpeed < 0) || (landingPointX2 < xMars
&& hSpeed > 0) || (Math.Abs(hSpeed) < (2 * maxHspeed)))
                {
                    Console.Error.Write($"not over Landing place. \nDesired Angle =
{desiredAngle}");
                    //mRotate = Convert.ToInt32(desiredAngle);
                    mRotate = Convert.ToInt32((xMars < landingPointX1) ? -desiredAngle
: (landingPointX2 < xMars) ? desiredAngle : 0);
                    Console.Error.WriteLine($"\nmRotate = {mRotate}");
                }
                else if (Math.Abs(hSpeed) > 2 * maxHspeed)
                {
                    mRotate = Convert.ToInt32((xMars < landingPointX1) ? (desiredAn-
gle): (landingPointX2 < xMars) ? (-desiredAngle) : 0);
                    Console.Error.WriteLine($"turn in right direction -> {mRotate}");

                }
            }
            else
            {
                Console.Error.WriteLine($"Landing Area");
                // Landing Area
                if (yMars < landingPointY + 200)
                {
                    Console.Error.WriteLine($"critical Area");
                    mRotate = 0;
                    mThrust = 4;
                }
                else if ((Math.Abs(hSpeed) <= (maxHspeed - 5)) && (Math.Abs(vSpeed) <=
(maxVspeed - 5)))
                {
                    Console.Error.WriteLine($"In Save Speed\n{hSpeed} vs {maxHspeed}\
n{vSpeed} vs {maxVspeed}");
                    mThrust = 2;
                }
                else
                {
                    mThrust = 4;
                    if (hSpeed != 0)
                    {
                        Console.Error.WriteLine($"correcting Rotation");
                        double sining = hSpeed / mSpeed;
                        double radRotate = Math.Atan(sining);
                        double degRotate = ConvertRadiansToDegrees(radRotate) * 100.0;
                        double mRotateTest = Math.Round(degRotate);
                        Console.Error.Write($"h={hSpeed} m={mSpeed} s={sining}
r={radRotate}\nd={degRotate} rot={mRotateTest}");
                        mRotate = Convert.ToInt32(mRotateTest);
                        mRotate = MaxValueAllowed(mRotate, 70, -70);
```

```
                    Console.Error.WriteLine($" mRot={mRotate}");
                }
                else
                    mRotate = 0;
            }
        }


        // Correct Min and Max Value
        MaxValueAllowed(Convert.ToInt32(mThrust),4,0);
        MaxValueAllowed(mRotate, 90, -90);



        // Write an action using Console.WriteLine()
        // To debug: Console.Error.WriteLine("Debug messages...");
        //
        // rotate power. rotate is the desired rotation angle. power is the de-
sired thrust power.
        Console.Error.WriteLine($"Output = {mRotate} {mThrust}");
        Console.WriteLine($"{mRotate} {mThrust}");
    }
}
public static double ConvertRadiansToDegrees(double radians)
{
    double degrees = (180.0 / Math.PI) * radians;
    return (degrees);
}
public static int MaxValueAllowed(int value, int Max, int Min)
{
    if (value > Max)
    {
        // Console.Error.WriteLine($"Max returned");
        return Max;
    }
    else if (value < Min)
    {
        // Console.Error.WriteLine($"Min returned");
        return Min;
    }
    // Console.Error.WriteLine($"Returning Value");
    return value;
}
}
```

2.1.2.2 Задание 2

Условие задачи представлено на рисунке 2.2.

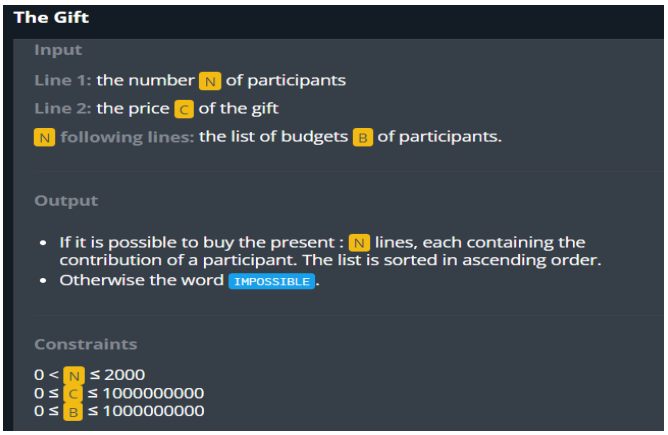


Рисунок 2.2

The Gift

```
using System;

internal class Solution
{
    private static void Main(string[] args)
    {
        var oodNumber = int.Parse(Console.ReadLine());
        var giftPrice = int.Parse(Console.ReadLine());

        // holds the budget for each ood
        var oodBudgets = new int[oodNumber];

        // total buget for all oods combined
        var totalBudget = 0;

        for (int i = 0; i < oodNumber; i++)
        {
            oodBudgets[i] = int.Parse(Console.ReadLine());
            totalBudget += oodBudgets[i];
        }

        if (totalBudget < giftPrice)
        {
            // the total budget is not enough to cover the cost of the gift
            Console.WriteLine("IMPOSSIBLE");
            return;
        }

        // if we have enough money, let's see how to best distribute the cost
        var oodsLeft = oodNumber;                           // keeps track of how
many oods still have monely left
        var oodPays = new int[oodNumber];                   // keeps track of how
much each ood will pay

        // as long as we haven't yet covered the cost of the gift,
        // and the cost requires more than 1 unit from each ood that still has
money left
        // .. keep splitting the cost ..
        while (giftPrice > oodsLeft)
```

```
            {
                var fair = giftPrice / oodsLeft;                // what would be an
ideal fair split given # of oods that still have money

                for (var i = oodNumber - 1; i >= 0; i--)
                {
                    if (oodBudgets[i] == 0) continue;           // this ood is out of
money, skip him..

                    var pays = Math.Min(oodBudgets[i], fair);   // how much will this
ood contribute?

                    oodBudgets[i] -= pays;                      // update his budget,
after payment
                    if (oodBudgets[i] == 0) oodsLeft--;         // if he's out of
money now, reduce # of ood left
                    giftPrice -= pays;                          // also reduce the re-
mainder cost

                    oodPays[i] += pays;                         // and update the to-
tal payed by this ood
                }
            }

            // if we're here, and there is still some cost left,
            // it must only require unit money from each ood left
            for (var i = oodNumber - 1; i >= 0 && giftPrice > 0; i--)
            {
                if (oodBudgets[i] == 0) continue;               // this ood is out of
money, skip'm..

                oodBudgets[i]--;                                // pays a unit..
                giftPrice--;                                    //   cost is reduced
                oodPays[i]++;                                   //   and update his
total amount to pay
            }

            // we have to list of how much each ood pays
            // but we have to present it in ascending order, so sort it..
            Array.Sort(oodPays);

            // we're done (just print out each payment)
            for (int i = 0; i < oodNumber; i++)
                Console.WriteLine(oodPays[i]);
        }
    }
```

2.2 Хранилища учетной информационной системы

2.2.1 Общие сведения

2.2.2 Индивидуальное задание

2.2.2.1 Задание 1

Условие задачи представлено на рисунке 2.3.

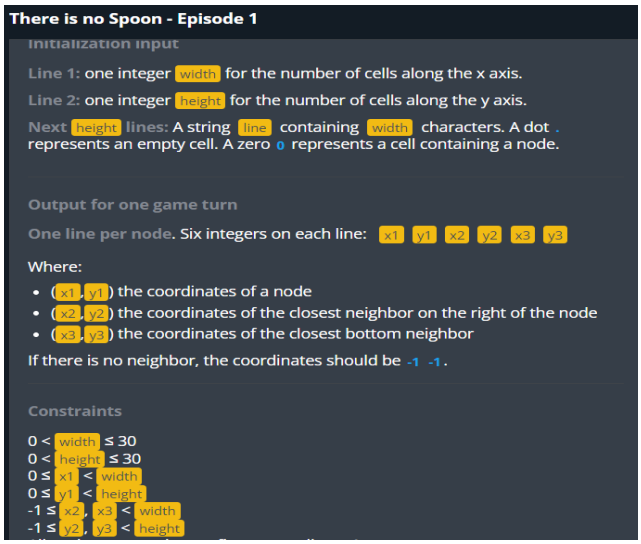


Рисунок 2.3

There is not spoon - episode 1

```
    using System;

    class Player
    {
        private static bool[,] matrixAlreadyFound;

        private static void FindSiblingCell(bool[,] matrix, int rowIndex, int
columnIndex)
        {
            matrixAlreadyFound[rowIndex, columnIndex] = true;
            Console.Write("{0} {1} ", columnIndex, rowIndex);

            var rightFind = false;
            int righIndex = columnIndex + 1;
            for (; righIndex < matrix.GetLength(1); righIndex++)
            {
                if (matrix[rowIndex, righIndex])
                {
                    rightFind = true;
                    break;
                }
            }

            var bottomFind = false;
            var bottomIndex = rowIndex + 1;
            for (; bottomIndex < matrix.GetLength(0); bottomIndex++)
            {
```

```
                if (matrix[bottomIndex, columnIndex])
                {
                    bottomFind = true;
                    break;
                }
            }

            if (!rightFind)
            {
                righIndex = -1;
                rowIndex = -1;
            }

            if (!bottomFind)
            {
                columnIndex = -1;
                bottomIndex = -1;
            }

            Console.Write(righIndex + " " + rowIndex + " ");
            Console.WriteLine(columnIndex + " " + bottomIndex);

            if (rightFind && !matrixAlreadyFound[rowIndex, righIndex])
            {
                FindSiblingCell(matrix, rowIndex, righIndex);
            }

            if (bottomFind && !matrixAlreadyFound[bottomIndex, columnIndex])
            {
                FindSiblingCell(matrix, bottomIndex, columnIndex);
            }
        }

        static void Main(string[] args)
        {
            var width = int.Parse(Console.ReadLine());
            var height = int.Parse(Console.ReadLine());
            var matrix = new bool[height, width];
            matrixAlreadyFound = new bool[height, width];

            for (int rowIndex = 0; rowIndex < height; rowIndex++)
            {
                var line = Console.ReadLine().ToCharArray();
                for (int columnIndex = 0; columnIndex < line.Length; columnIndex++)
                {
                    matrix[rowIndex, columnIndex] = line[columnIndex] == '.' ? false :
true;
                }
            }

            for (int rowIndex = 0; rowIndex < matrix.GetLength(0); rowIndex++)
            {
                for (int columnIndex = 0; columnIndex < matrix.GetLength(1); columnIn-
dex++)
                {
                    if (matrix[rowIndex, columnIndex] && !matrixAlreadyFound[rowIndex,
columnIndex])
```

```
                    FindSiblingCell(matrix, rowIndex, columnIndex);
                }
            }
        }
    }
```

2.2.2.2 Задание 2

Условие задачи представлено на рисунке 2.4.

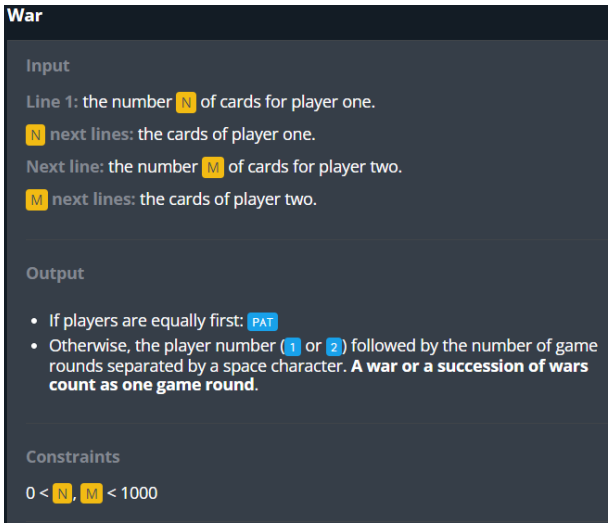


Рисунок 2.4

WAR

```
using System;
using System.Collections.Generic;
using System.Linq;

public class Solution
{
    public static int GetCardValue(string card)
    {
        if (int.TryParse(card.Substring(0, card.Length - 1), out var val))
            return val;
        else
        {
            if (card[0] == 'J')
                return 11;
            else if (card[0] == 'Q')
                return 12;
            else if (card[0] == 'K')
                return 13;
            else if (card[0] == 'A')
                return 14;
        }
        throw new InvalidOperationException();
    }

    public static int GetWinner(string card1, string card2)
    {
        if (GetCardValue(card1) > GetCardValue(card2))
```

```
            return 1;
        else if (GetCardValue(card1) < GetCardValue(card2))
            return 2;
        else
            return 0;
    }

    public static void Main(string[] args)
    {
        LinkedList<string> cardsP1 = new LinkedList<string>();
        LinkedList<string> cardsP2 = new LinkedList<string>();
        LinkedList<string> cardsOnTableP1 = new LinkedList<string>();
        LinkedList<string> cardsOnTableP2 = new LinkedList<string>();


        int n = int.Parse(Console.ReadLine()); // the number of cards for player 1
        for (int i = 0; i < n; i++)
        {
            string cardp1 = Console.ReadLine(); // the n cards of player 1
            cardsP1.AddLast(cardp1);
        }
        int m = int.Parse(Console.ReadLine()); // the number of cards for player 2
        for (int i = 0; i < m; i++)
        {
            string cardp2 = Console.ReadLine(); // the m cards of player 2
            cardsP2.AddLast(cardp2);
        }

        int rounds = 0;
        while (true)
        {
            // The game ends when one player no longer has cards
            if (!cardsP1.Any())
            {
                Console.WriteLine($"2 {rounds}");
                return;
            }
            else if (!cardsP2.Any())
            {
                Console.WriteLine($"1 {rounds}");
                return;
            }

            bool playingWar;
            do
            {
                playingWar = false;

                // Each player draws a card
                cardsOnTableP1.AddLast(cardsP1.First.Value);
                cardsOnTableP2.AddLast(cardsP2.First.Value);
                cardsP1.RemoveFirst();
                cardsP2.RemoveFirst();

                // Play the round
                int roundWinner = GetWinner(cardsOnTableP1.Last.Value, card-
sOnTableP2.Last.Value);
```

```
                    if (roundWinner == 1)
                    {
                        foreach (var card in cardsOnTableP1)
                        {
                            cardsP1.AddLast(card);
                        }

                        foreach (var card in cardsOnTableP2)
                        {
                            cardsP1.AddLast(card);
                        }

                        cardsOnTableP1.Clear();
                        cardsOnTableP2.Clear();
                    }
                    else if (roundWinner == 2)
                    {
                        foreach (var card in cardsOnTableP1)
                        {
                            cardsP2.AddLast(card);
                        }

                        foreach (var card in cardsOnTableP2)
                        {
                            cardsP2.AddLast(card);
                        }

                        cardsOnTableP1.Clear();
                        cardsOnTableP2.Clear();
                    }
                    else // war!
                    {
                        // If we don't have enough cards => draw
                        if ((cardsP1.Count < 4) || (cardsP2.Count < 4))
                        {
                            Console.WriteLine("PAT");
                            return;
                        }

                        // Place 3 cards face down
                        for (int i = 0; i < 3; ++i)
                        {
                            cardsOnTableP1.AddLast(cardsP1.First.Value);
                            cardsOnTableP2.AddLast(cardsP2.First.Value);
                            cardsP1.RemoveFirst();
                            cardsP2.RemoveFirst();
                        }

                        // Draw the next cards
                        playingWar = true;
                    }
                } while (playingWar);

                rounds++;
            }
        }
    }
```

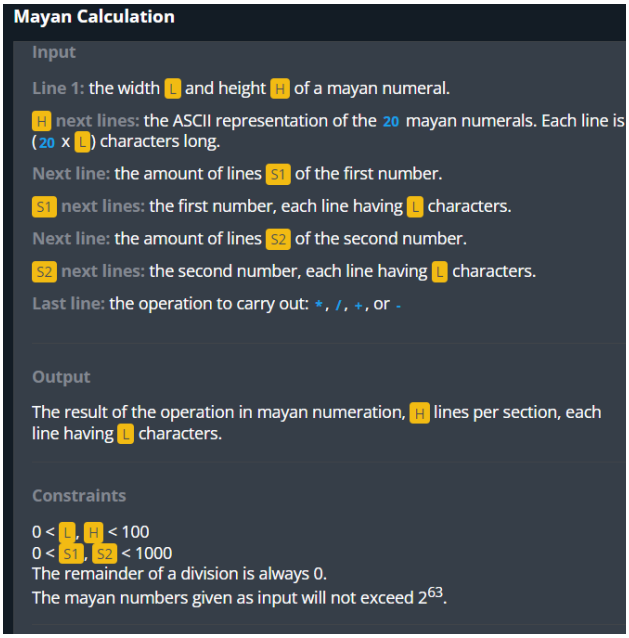
2.3 Способы отслеживания ошибок в учетной информационной системе

2.3.1 Общие сведения

2.3.2 Индивидуальное задание

2.3.2.1 Задание 1

Условие задачи представлено на рисунке 2.5.



Mayan Calculation

Input

Line 1: the width L and height H of a mayan numeral.

H next lines: the ASCII representation of the 20 mayan numerals. Each line is (20 x L) characters long.

Next line: the amount of lines S1 of the first number.

S1 next lines: the first number, each line having L characters.

Next line: the amount of lines S2 of the second number.

S2 next lines: the second number, each line having L characters.

Last line: the operation to carry out: *, /, +, or -

Output

The result of the operation in mayan numeration, H lines per section, each line having L characters.

Constraints

0 < L, H < 100
0 < S1, S2 < 1000
The remainder of a division is always 0.
The mayan numbers given as input will not exceed 2^{63}.

Рисунок 2.5

Mayan Calculation

```
using System;
using System.Collections.Generic;
using System.Linq;



public class Solution
{
    public class Number
    {
        public Number(long number, IList<string> representation)
        {
            NumberInBase10 = number;
            matrix = representation;
```

```
            }

            public IList<string> matrix { get; set; }

            public long NumberInBase10 { get; private set; }

            public string GetRepresentation()
            {
                return string.Concat(matrix);
            }
        }

        public static long GetNumber(int H, IList<Number> numbers)
        {
            var firstNumbersRep = new List<string>();
            int S1 = int.Parse(Console.ReadLine());
            for (int i = 0; i < S1; i++)
            {
                firstNumbersRep.Add(Console.ReadLine());
            }

            var numberOfNumber = S1 / H;
            var calPuis = numberOfNumber - 1;
            var result = 0;
            for (int i = 0; i < numberOfNumber; i++)
            {
                var matrix = new List<string>();
                for (int j = 0; j < H; j++)
                {
                    matrix.Add(string.Concat(firstNumbersRep[(i * H) + j] + Environ-
ment.NewLine));
                }
                result += Convert.ToInt32(numbers.Where(x =>
x.matrix.SequenceEqual(matrix)).First().NumberInBase10 * Math.Pow(20, calPuis--));
            }
            return Convert.ToInt32(result);
        }

        public static void Main(string[] args)
        {
            var numbers = new List<Number>();
            string[] inputs = Console.ReadLine().Split(' ');
            int L = int.Parse(inputs[0]);
            int H = int.Parse(inputs[1]);
            var listRepresentation = new List<string>();
            for (int i = 0; i < H; i++)
            {
                listRepresentation.Add(Console.ReadLine());
            }

            var numberOfNumber = listRepresentation.First().Length / L;
            for (int i = 0; i < numberOfNumber; i++)
            {
                var matrix = new List<string>();
                for (int j = 0; j < H; j++)
                {
```

```
                    matrix.Add(string.Concat(listRepresentation[j].Skip(i *
L).Take(L)) + Environment.NewLine);
                }
                numbers.Add(new Number(i, matrix));
            }

            var first = GetNumber(H, numbers);

            var second = GetNumber(H, numbers);

            string operation = Console.ReadLine();
            long resultOperation = 0;

            switch (operation)
            {
                case "+":
                    resultOperation = first + second;
                    break;

                case "-":
                    resultOperation = first - second;
                    break;

                case "/":
                    resultOperation = first / second;
                    break;

                case "*":
                    resultOperation = first * second;
                    break;

                default:
                    break;
            }

            var resultBase20 = Encode(resultOperation, 20);
            foreach (var num in resultBase20)
            {
                Console.Write(numbers.Where(x => x.NumberInBase10 == num).First().Ge-
tRepresentation());
            }
        }

        private static IEnumerable<long> Encode(long input, int baseToConvert)
        {
            if (input < 0) throw new ArgumentOutOfRangeException("input", input, "in-
put cannot be negative");

            var result = new Stack<long>();
            if (input == 0) result.Push(0);

            while (input != 0)
            {
                result.Push(input % baseToConvert);
                input /= baseToConvert;
            }
            return result;
```

```
    }
}
```

2.3.2.2 Задание 2

Условие задачи представлено на рисунке 2.6.

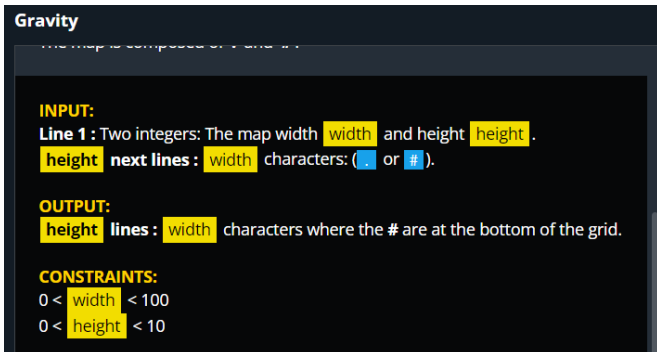


Рисунок 2.6

Gravity

```
using System;

 public class Solution
 {
     public static void Main(string[] args)
     {
         string[] inputs = Console.ReadLine().Split(' ');
         int width = int.Parse(inputs[0]);
         int height = int.Parse(inputs[1]);
         var result = new int[width];

         for (int i = 0; i < height; i++)
         {
             var line = Console.ReadLine().ToCharArray();
             for (int j = 0; j < width; j++)
             {
                 if (line[j] == '#')
                 {
                     result[j] += 1;
                 }
             }
         }

         for (int i = 0; i < height; i++)
         {
             for (int j = 0; j < width; j++)
             {
                 if (result[j] >= height - i)
                 {
                     Console.Write("#");
                 }
                 else
                 {
```

```
                    Console.Write(".");
                }
            }

            Console.WriteLine("");
        }
    }
}
```

3 СОПРОВОЖДЕНИЕ ОБЛАЧНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

3.1 Отличия серверных и облачных технологий

3.1.1 Общие сведения

3.1.2 Индивидуальное задание

3.1.2.1 Задание 1

Условие задачи представлено на рисунке 3.1.

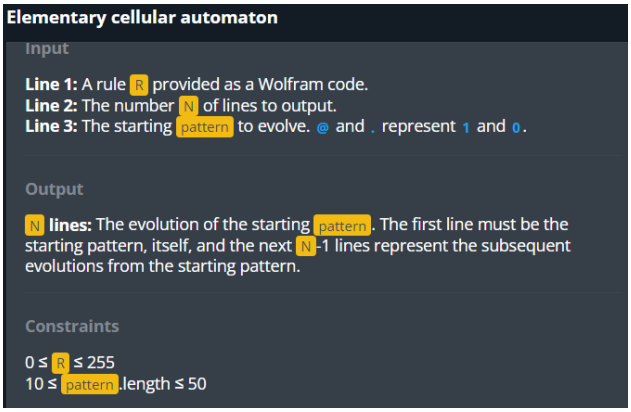


Рисунок 3.1

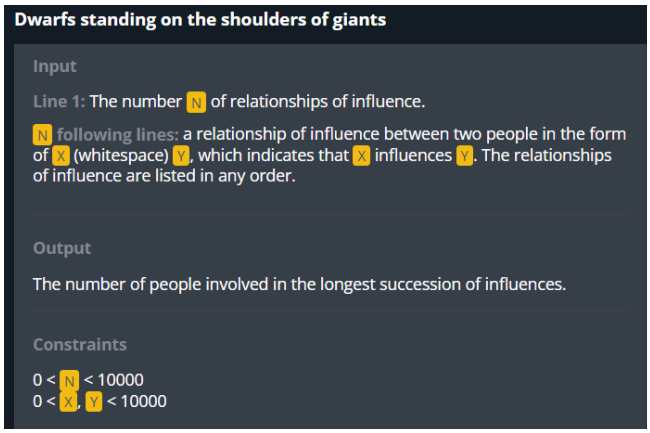
Elementary cellular automaton

```
using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;

    public class Solution
    {
        private static string GetEvolution(string pattern, Dictionary<string, char>
neighborhoods)
        {
            var result = new StringBuilder();

            var neighborhoodFirstKey = pattern.Last() +
string.Concat(pattern.Skip(0).Take(2));
            result.Append(neighborhoods[neighborhoodFirstKey].ToString());

            for (int i = 1; i < pattern.Length - 1; i++)
            {
                var neighborhoodKey = string.Concat(pattern.Skip(i - 1).Take(3));
                result.Append(neighborhoods[neighborhoodKey].ToString());
            }

            var neighborhoodLastKey = string.Concat(pattern.Skip(pattern.Length -
2).Take(2)) + pattern.First();
            result.Append(neighborhoods[neighborhoodLastKey].ToString());
```

```
            return result.ToString();
        }

        public static void Main(string[] args)
        {
            var wolframCode = Convert.ToString(int.Parse(Console.ReadLine()), 2).Re-
place('0', '.').Replace('1', '@').PadLeft(8, '.');
            var numberOfEvolution = int.Parse(Console.ReadLine());
            var pattern = Console.ReadLine();

            var neighborhoods = new Dictionary<string, char>
        {
            { "@@@", wolframCode.ElementAt(0) },
            { "@@.", wolframCode.ElementAt(1) },
            { "@.@", wolframCode.ElementAt(2) },
            { "@..", wolframCode.ElementAt(3) },
            { ".@@", wolframCode.ElementAt(4) },
            { ".@.", wolframCode.ElementAt(5) },
            { "..@", wolframCode.ElementAt(6) },
            { "...", wolframCode.ElementAt(7) }
        };

            for (int i = 0; i < numberOfEvolution; i++)
            {
                Console.WriteLine(pattern);
                pattern = GetEvolution(pattern, neighborhoods);
            }
        }
    }
```

3.1.2.2 Задание 2

Условие задачи представлено на рисунке 3.2.



Dwarfs standing on the shoulders of giants

Input

Line 1: The number N of relationships of influence.

N following lines: a relationship of influence between two people in the form of X (whitespace) Y, which indicates that X influences Y. The relationships of influence are listed in any order.

Output

The number of people involved in the longest succession of influences.

Constraints

0 < N < 10000
0 < X, Y < 10000

Рисунок 3.2

Dwarfs standing on the shoulders of giants

```
using System;
using System.Collections.Generic;
using System.Linq;

public class Solution
```

```
    {
        public static void Main(string[] args)
        {
            var relationships = new Dictionary<int, List<int>>();
            var numberOfRelationships = int.Parse(Console.ReadLine());

            for (int i = 0; i < numberOfRelationships; i++)
            {
                string[] inputs = Console.ReadLine().Split(' ');
                var influencer = int.Parse(inputs[0]);
                var influenced = int.Parse(inputs[1]);
                AddRelationship(relationships, influencer, influenced);
            }

            var longestRelationship = relationships.Keys
                                        .Select(influencer => 1 + RouteRelationships(re-
lationships, influencer))
                                        .Max();

            Console.WriteLine(longestRelationship);
        }

        private static int RouteRelationships(Dictionary<int, List<int>> relation-
ships, int influencer)
        {
            var longestRelationship = 0;
            if (relationships.ContainsKey(influencer))
            {
                longestRelationship = relationships[influencer]
                                        .Select(influenced => 1 + RouteRelationships(re-
lationships, influenced))
                                        .Max();
            }
            return longestRelationship;
        }

        private static void AddRelationship(Dictionary<int, List<int>> relationships,
int influencer, int influenced)
        {
            if (!relationships.ContainsKey(influencer))
            {
                relationships.Add(influencer, new List<int>());
            }
            relationships[influencer].Add(influenced);
        }
    }
```

3.2 Основные виды облачных архитектур

3.2.1 Общие сведения

3.2.2 Индивидуальное задание

3.2.2.1 Задание 1

Условие задачи представлено на рисунке 3.3.

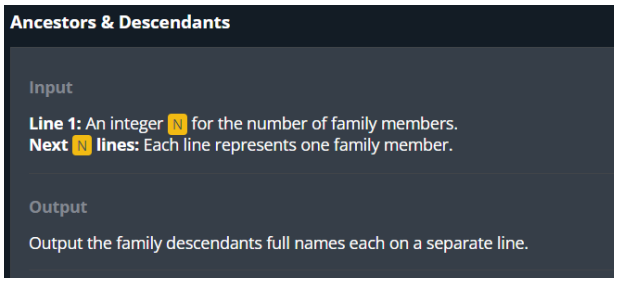


Рисунок 3.3

Ancestors and Descendants

```
using System;
using System.Collections.Generic;
using System.Linq;

public class Solution
{
    private static void Main(string[] args)
    {
        var list = new List<Node>();
        int count = int.Parse(Console.ReadLine());

        for (int i = 0; i < count; i++)
        {
            string line = Console.ReadLine();
            if (!line.StartsWith("."))
                list.Add(new Node { Id = line });
            else
                list.Last().AddChild(line.Substring(1));
        }

        foreach (var item in list)
        {
            foreach (var s in item.GetChildrenHierarchy())
            {
                Console.WriteLine(s);
            }
        }
    }
}

public class Node
{
```

```
        public Node()
        {
            Children = new List<Node>();
        }


        public string Id { get; set; }
        public List<Node> Children { get; set; }

        public void AddChild(string name)
        {
            if (name.StartsWith("."))
            {
                name = name.Substring(1);
                Children.Last().AddChild(name);
            }
            else
            {
                Children.Add(new Node { Id = name });
            }
        }

        public IEnumerable<string> GetChildrenHierarchy()
        {
            var result = Id + " > ";
            var list = new List<string>();

            if (Children.Count() == 0)
                yield return Id;
            else
            {
                foreach (var itesm in Children)
                {
                    if (itesm.Children.Count() == 0)
                    {
                        yield return result + itesm.Id;
                    }
                    else
                    {
                        foreach (var item in itesm.GetChildrenHierarchy())
                        {
                            yield return result + item;
                        }
                    }
                }
            }
        }
    }
```

3.2.2.2 Задание 2

Условие задачи представлено на рисунке 3.4.

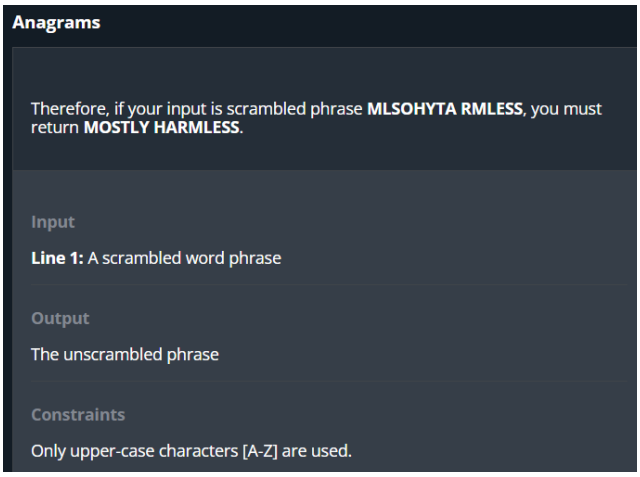


Рисунок 3.4

Anagrams

```
    using System;
    using System.Collections.Generic;
    using System.Linq;

    public class Solution
    {
        public static void Main(string[] args)
        {
            var phrase = Phase4(Console.ReadLine());
            phrase = Phase3(phrase);
            phrase = Phase2(phrase);
            Console.WriteLine(Phase1(phrase));
        }

        public static string Phase1(string phrase)
        {
            // 2nd letter of the alphabet
            var letters = Enumerable.Range(0, 26).Where(i => i % 2 != 0).Select(i =>
(char)(i + 65));
            var queue = new Queue<char>();

            foreach (var character in phrase.Reverse().Where(c =>
letters.Contains(c)))
            {
                queue.Enqueue(character);
            }

            return Replace(phrase, letters, queue);
        }

        public static string Phase2(string phrase)
        {
            // 3rd letter of the alphabet
            var letters = Enumerable.Range(0, 26).Where(i => i % 3 == 2).Select(i =>
(char)(i + 65));
            var queue = new Queue<char>();

            var ph1 = phrase.Where(c => letters.Contains(c));
            foreach (var character in ph1.Skip(1))
```

```
            {
                queue.Enqueue(character);
            }
            if (ph1.Count() >= 2)
                queue.Enqueue(ph1.First());

            return Replace(phrase, letters, queue);
        }

        public static string Phase3(string phrase)
        {
            // 4th letter of the alphabet
            var letters = Enumerable.Range(0, 26).Where(i => i % 4 == 3).Select(i =>
(char)(i + 65));
            var queue = new Queue<char>();

            var ph2 = phrase.Where(c => letters.Contains(c));

            if (ph2.Count() >= 2)
            {
                queue.Enqueue(ph2.Last());
                foreach (var character in ph2.Take(ph2.Count() - 1))
                {
                    queue.Enqueue(character);
                }
            }

            return Replace(phrase, letters, queue);
        }

        public static string Phase4(string phrase)
        {
            // 4th letter of the alphabet
            var wordLength = phrase.Split(" ".ToCharArray()).Select(c => c.Length).Re-
verse();
            phrase = phrase.Replace(" ", string.Empty);
            var result = string.Empty;

            foreach (var length in wordLength)
            {
                result += phrase.Substring(0, length) + " ";
                phrase = phrase.Substring(length);
            }

            return result.Trim();
        }
```

4 ДОКУМЕНТИРОВАНИЕ ПРОЦЕССОВ ВНЕДРЕНИЯ И СОПРОВОЖДЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

4.1 Жизненный цикл информационной системы

4.1.1 Общие сведения

4.1.2 Индивидуальное задание

4.1.2.1 Задание 1

Условие задачи представлено на рисунке 4.1.

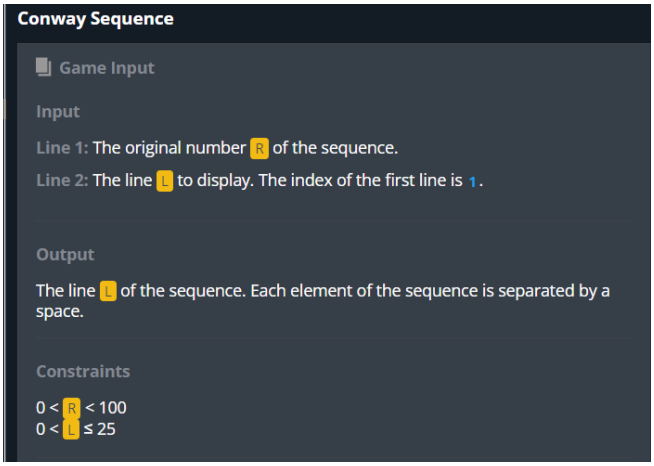


Рисунок 4.1

Conway Sequence

```
using System;
using System.Linq;
public class Solution
{
public static string GetDescription(string valuesString) // передаем значение
{
var values = valuesString.Split(' ').Select(x => int.Parse(x)).ToList(); // входящую
сроку разделяем на элементы
var lastVal = values.First(); // переносим самый первый элемент последовательности
var count = 0;
var result = string.Empty; // пустая строка

for (int i = 0; i < values.Count; i++) // до последнего элемента в values
{
if (lastVal != values[i]) // смотрим равен ли предыдущий эл-т текущему
{
result += count + " " + lastVal + " ";
count = 1; // увеличиваем на 1
lastVal = values[i]; // запоминаем как текущий эл-т
}
else count++; // если они равны, то счетчик увеличиваем на 1
```

```
if (i + 1 == values.Count) // проверяем не дошли ли мы до последнего эл-та
{
result += count + " " + values[i]; // если след. эл-т последний, то прибавляем к
результату добавляем счетчик и через пробел добавляем знач. последнего элемента
}
}
return result.Trim(); // возврат результата, Trim - убирает эл-ты
}
public static void Main(string[] args)
{
var R = Console.ReadLine(); // исходный номер последовательности, начало
int L = int.Parse(Console.ReadLine()); // строка, последовательность,которой нужно
вывести
for (int i = 0; i < L - 1; i++)
{
R = GetDescription(R); // вызываем метод и передаем значение каждой строки
}
Console.WriteLine(R); // вывод
}
}
```

4.1.2.2 Задание 2

Условие задачи представлено на рисунке 4.2.

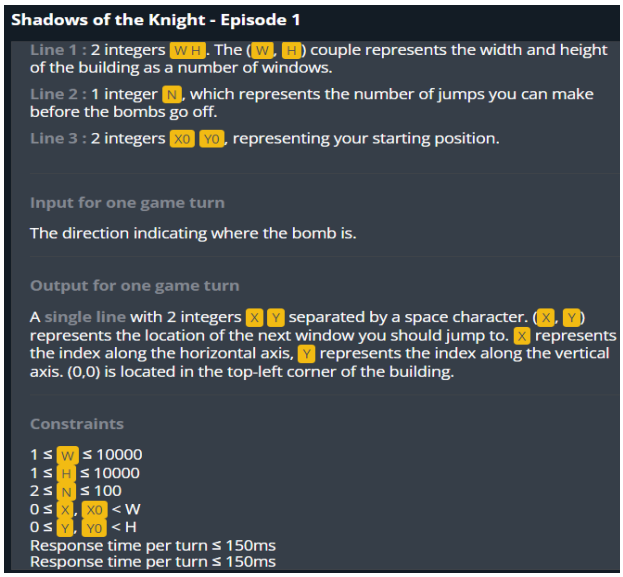


Рисунок 4.2

Shadow of knight-ep1

```
using System;
using System.Linq;
using System.IO;
using System.Text;
using System.Collections;
using System.Collections.Generic;


class Player
```

```
{
    static void Main(string[] args)
    {
        string[] inputs;//массив
        inputs = Console.ReadLine().Split(' ');//разделяем входящую строчку на 2
массива
        int W = int.Parse(inputs[0]); // ширина здания.
        int H = int.Parse(inputs[1]); // высота здания.
        int W0 = 0;//начальная позиция
        int H0 = 0;//начальная позиция
        int N = int.Parse(Console.ReadLine()); // максимальное количество ходов до
окончания игры.
        inputs = Console.ReadLine().Split(' ');//строка с координатами бэтмана и
разделяекм на пробелы
        int X0 = int.Parse(inputs[0]);//координаты
        int Y0 = int.Parse(inputs[1]);//координаты

        // игровой цикл
        while (true)
        {
            string bombDir = Console.ReadLine(); // направление бомб от текущего
местоположения Бэтмена (U, UR, R, DR, D, DL, L или UL)

            if (bombDir[0] == 'U')//напровление вверх, значит рассматриваем что снизу
            {
                H = Y0;//
                Y0 = H0 + ((Y0 - H0) / 2);
            }
            else if (bombDir[0] == 'D')//напровление вниз, значит рассматриваем что
вверху
            {
                H0 = Y0;
                Y0 = H - ((H - Y0) / 2);
            }

            if (bombDir[0] == 'L' || bombDir.Length > 1 && bombDir[1] ==
'L')//направление влево, значит рассматриваем что справо
            {
                W = X0;
                X0 = W0 + ((X0 - W0) / 2); //((X0 - W0) / 2);
            }
            else if (bombDir[0] == 'R' || bombDir.Length > 1 && bombDir[1] ==
'R')//направлние вправо, значит рассматриваем что слева
            {
                W0 = X0;
                X0 = W - ((W - X0) / 2);
            }

            //местоположение следующего окна, в которое должен прыгнуть Бэтмен.
            Console.WriteLine(X0 + " " + Y0);
            Console.Error.WriteLine(W0);
        }
    }
}
```

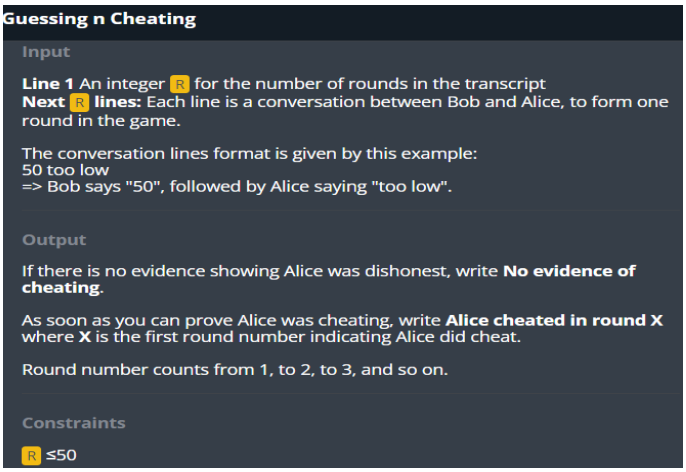
4.2 Стандарты документирования информационных систем

4.2.1 Общие сведения

4.2.2 Индивидуальное задание

4.2.2.1 Задание 1

Условие задачи представлено на рисунке 4.3.



Guessing n Cheating

Input

Line 1 An integer R for the number of rounds in the transcript
Next R lines: Each line is a conversation between Bob and Alice, to form one round in the game.

The conversation lines format is given by this example:
50 too low
=> Bob says "50", followed by Alice saying "too low".

Output

If there is no evidence showing Alice was dishonest, write **No evidence of cheating**.

As soon as you can prove Alice was cheating, write **Alice cheated in round X** where **X** is the first round number indicating Alice did cheat.

Round number counts from 1, to 2, to 3, and so on.

Constraints

R ≤50

Рисунок 4.3

Guessing and Cheating

```
using System;
using System.Linq;
using System.IO;
using System.Text;
using System.Collections;
using System.Collections.Generic;

public class Solution
{
public static void Main()
{
int R = int.Parse(Console.ReadLine()); // количество раундов, целое число
int start = 1, end = 100; // диапазон загаданных чисел
for (int i = 0; i < R; i++)
{
string[] guess = Console.ReadLine().Split(); // создаем массив, считываем, что говорит
боб и алиса через консоль
int bob = int.Parse(guess[0]); // запоминаем, что сказал боб (число)
string alice = guess[1] + " " + guess[2]; // запоминаем, что сказала алиса (фразы с
пробелами)
if (alice == "too low") // если алиса сказала "слишком низко"
{
start = Math.Max(start, bob + 1); // вырезаем то, что ниже зачения боба, Math.Max - из
двух входящих значений выбираем максимальное
}
else if (alice == "too high") // если алиса сказала "слишком высоко"
```

```
{
end = Math.Min(end, bob - 1); // вырезаем то, что выше значения боба, Math.Min - из
двух входящих значений выбираем минимальное
}
if (end < start || alice == "right on" && !(start <= bob && bob <= end)) // Проверяем,
не обманули ли Алису, || - или, && - и, ! - не
{
Console.WriteLine("Alice cheated in round {0}", i + 1); // Алиса обманула в раунде
Environment.Exit(0);
}
}
Console.WriteLine("No evidence of cheating"); // Нет доказательств обмана
}
}
```

ЗАКЛЮЧЕНИЕ

В результате прохождения учебной практики были рассмотрены следующие вопросы: рассмотрение вопросов состава информационных систем; рассмотрение вопросов классификации информационных систем; рассмотрение вопросов функционирования облачных информационных систем; изучение архитектуры учетной информационной системы; изучение хранилища учетной информационной системы; изучение способов отслеживания ошибок в учетной информационной системе; изучение способов исправления ошибок в учетной информационной системе; рассмотрение вопросов отличия серверных и облачных технологий; изучение основных видов облачных архитектур; изучение способов мониторинга облачных сервисов; изучение жизненного цикла информационной системы; изучение стандартов документирования информационных систем; составление календарного графика внедрения информационной системы.

СПИСОК ЛИТЕРАТУРЫ

1. Перлова, О. Н. Соадминистрирование баз данных и серверов : учебник для студентов среднего профессионального образования по специальности 09.02.07 "Информационные системы и программирование" / О. Н. Перлова, О. П. Ляпина ; О. Н. Перлова, О. П. Ляпина. – Москва : Академия, 2018. – 304 с. с. – URL: http://academia-moscow.ru/catalogue/4831/345911/ (дата обращения: 09.06.2021). – Текст : электронный.

2. Боровская, Е. В. Основы искусственного интеллекта / Е. В. Боровская, Н. А. Давыдова. – Москва : Лаборатория знаний, 2020. – 130 с. – ISBN 9785001019084. – URL: http://biblioclub.ru/index.php?page=book_red&id=595419 (дата обращения: 06.06.2021). – Текст : электронный.

3. Рудаков, А. В. Технология разработки программных продуктов : учебник для студентов учреждений среднего профессионального образования, обучающихся по специальности "Программное обеспечение вычислительной техники и автоматизированных систем" : [профессиональный модуль ПМ.03 "Участие в интеграции программных модулей" (МДК.03.01)] / А. В. Рудаков ; А. В. Рудаков. – 11-е изд., стер. – Москва : Академия, 2017. – 208 с. с. – URL: http://academia-moscow.ru/catalogue/4831/362819/ (дата обращения: 09.06.2021). – Текст : электронный.

4. Гохберг, Г. С. Информационные технологии : учебник для образовательных организаций, реализующих программы среднего профессионального образования по специальностям "Информационные системы и программирование", "Сетевое и системное администрирование" / Г. С. Гохберг, А. В. Зафиевский, А. А. Короткин ; Г. С. Гохберг, А. В. Зафиевский, А. А. Короткин. – Москва : Академия, 2017. – 240 с. с. – (Профессиональное образование). – URL:

http://academia-moscow.ru/catalogue/4831/297236/. – Текст : непосредственный + электронный.

5. Компьютерные сети : учебник для среднего профессионального образования по специальностям 09.02.06 "Сетевое и системное администрирование", 09.02.07 "Информационные системы и программирование" / В. В. Баринов, И. В. Баринов, А. В. Пролетарский, А. Н. Пылькин ; В. В. Баринов, И. В. Баринов, А. В. Пролетарский, А. Н. Пылькин. – Москва : Академия, 2018. – 192 с. с. – URL: http://academia-moscow.ru/catalogue/4831/345920/ (дата обращения: 09.06.2021). – Текст : электронный.

6. Куприянов, Д. В. Информационное обеспечение профессиональной деятельности.: учебник и практикум для СПО / Куприянов Д. В.. – Москва : Юрайт, 2020. – 255 с. – ISBN 978-5-534-00973-6. – URL: https://urait.ru/book/informacionnoe-obespechenie-professionalnoy-deyatelnosti-451935 (дата обращения: 06.06.2021). – Текст : электронный.

7. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : Учебное пособие / Г. Н. Федорова. – Москва : НИЦ ИНФРА-М, 2020. – 336 с. – ISBN 978-5-906818-41-6. – URL: http://znanium.com/go.php?id=1047718 (дата обращения: 06.06.2021). – Текст : электронный.

8. Учебная практика УП.06.01 : методические материалы для обучающихся специальности СПО 09.02.07 "Информационные системы и программирование" очной формы обучения / ФГБОУ ВО "Кузбас. гос. техн. ун-т им. Т. Ф. Горбачева", Каф. информ. и автоматизир. произв. систем ; сост. С. А. Асанов. – Кемерово : КузГТУ, 2018. – 14 с. – URL: http://library.kuzstu.ru/meto.php?n=9276 (дата обращения: 09.06.2021). – Текст : электронный.