Модул 2. Програмиране

Тема 1. Системи за контрол на версиите

Задача 1.1. Създайте GitHub профил

Създайте GitHub профил: http://github.com

Задача 1.2. Създайте GitHub хранилище

- **Създайте GitHub хранилище: https://github.com/new.**
- **Клонирайте хранилището на работния плот.**
- **Създайте файл README.md със съдържание First Repository.**
- **Качете промените в хранилището.**

Примерно решение в конзолата

```
git clone https://github.com/dimitarminchev/RepositoryName.git
cd RepositoryName
echo "First Repository" >> README.md
git add README.md
git commit -m "First Commit"
git push
```

Задача 1.3. Клонирайте хранилището два пъти

Клонирайте създаденото хранилище на две различни места на вашето устройство.

Примерно решение в конзолата

```
mkdir repo1
cd repo1
git clone https://github.com/dimitarminchev/RepositoryName.git
cd ..
mkdir repo2
cd repo2
git clone https://github.com/dimitarminchev/RepositoryName.git
```

Задача 1.4. Създайте конфликт

Променете съдържанието на двете папки по различен начин:

- **В repo1/RepositoryName отворете файла README.md и добавете ред: "Update 1"**
- **В repo2/RepositoryName отворете файла README.md и добавете ред: "Update 2"**

Задача 1.5. Качете промените от копието repo1

Подсказка

Може да използвате командата git commit, като напишете кратък коментар в кавички след опцията -m, описващ каква е промяната.

Примерно решение в конзолата

```
cd .\repo1\RepositoryName\
```

```
git add README.md
git commit -m "Update 1"
git push
```

Задача 1.6. Опитайте сега да обновите вашето repo2 копие

Отворете папка repo2 и в конзолата изпълнете следните команди:

- Добавете променения файл: git add README.md
- Предайте вашите промени, заедно с коментар за промените: git commit -m "Update 2"
- Обновете локалното ви хранилище: git pull

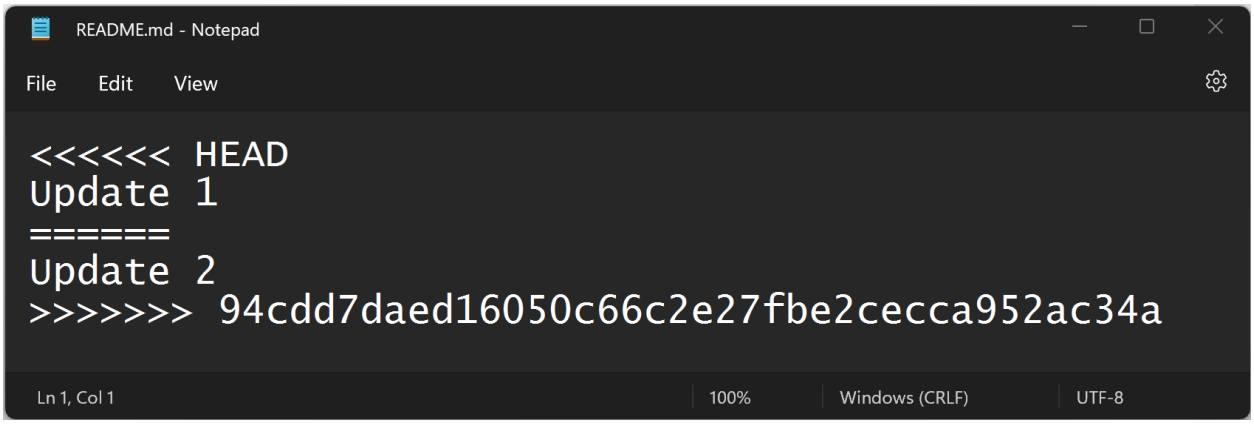
Примерно решение в конзолата

```
cd .\repo2\RepositoryName\
git add .\README.md
git commit -m "Update 2"
git pull
```

```
PS C:\Users\mitko\Desktop> cd .\repo2\RepositoryName\
PS C:\Users\mitko\Desktop\repo2\RepositoryName> git add .\README.md
PS C:\Users\mitko\Desktop\repo2\RepositoryName> git commit -m "Update 2"
[main fe8b776] Update 2
 1 file changed, 2 insertions(+), 1 deletion(-)
PS C:\Users\mitko\Desktop\repo2\RepositoryName> git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 258 bytes | 8.00 KiB/s, done.
From https://github.com/dimitarminchev/RepositoryName
   e7fc6ef..1247204  main       -> origin/main
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

Задача 1.7. Сега имаме конфликт при сливането, който трябва да разрешим.

- Отворете файла README.md във вашето repo2 копие, трябва да е нещо подобно на:



- **<<<<<<< HEAD** маркира началото на локалната версия на файла; **=======** разделя локалната версия от тази в хранилището. **>>>>>>>** маркира края на файла и след него е записан номера commit. За да разрешите конфликта, трябва да изтриете всичките три специални маркера и да изберете коя версия на файла да запазите. Имате три възможности:
    - Можете да изтриете **Update 1** или **Update 2** (т.е. единия от двата различни текста, породили конфликта);
    - Можете да запазите и двата текста и да изтриете само маркерите;
    - Или можете да напишете напълно различно, ново изречение.
- Сега, след като сте разрешили конфликта, покажете променения файл **git add .**, публикувайте пак промените **git commit -m "Conflict Resolving"** и синхронизирайте вашите промени с отдалеченото хранилище **git sync**.

Задача 1.8. Екипна работа

Работа в екип от (около) 5 ученика в клас. Всеки екип трябва да си избере „ръководител на екипа“. Ръководителят създава организацията в GitHub:

- Нова организация от: https://github.com/settings/profile, страница *Organizations*.
    - Избира уникално име за организацията, например **TestOrg** и добавя членове в нея.
- След това създава хранилище, например **TestRepo.**

Задача 1.9. Добавете файл към GitHub

Членовете на екипа (включително и ръководителя на екипа) добавят няколко файла:

1. Клонирайте **TestRepo** на вашия компютър.
2. Създайте нов файл във вашата работна директория.
    - Наречете файла <вашето_име>.txt
3. Добавете някакъв текст в него, например „Казвам се ....“
4. Commit-нете новия файл във вашето локално хранилище.
5. Синхронизирайте промените за да изпратите файла в отдалеченото хранилище.
6. Разгледайте хранилището на адрес https://github.com/user/repo, за да проверите дали файлът ви е успешно изпратен в GitHub.

Задача 1.10. Създайте конфликт в Git и слейте промените

- Нека всички членове на екипа създадат общ файл config.txt
- Всеки член на екипа да добави някакви настройки във файла config.txt, например
    - *Име = Петър*
    - *размер = 100*
    - *email = peter@dir.bg*
- Всеки член на екипа commit-ва своите локални промени.
- Всеки член на екипа синхронизира своите промени с общото хранилище.
    - При първия това ще стане успешно, без конфликти.
    - Другите ще имат конфликт, промените от който трябва да бъдат слети.
    - Разрешете конфликта:
        - Редактирайте слетите промени + commit и синхронизиране отново.

Тема 2. Типове данни

Задача 2.1. Преобразуване в различни бройни системи

- **Преобразувайте 1234 (10) в двоична и шестнадесетична бройна системи.**
- **Преобразувайте 1100101 (2) в десетична и шестнадесетична бройна системи.**
- **Преобразувайте ABC (16) в десетично и двоична бройна системи.**

Решение

```
namespace NumeralSystemsConverter
{
    class Program
```

```
{
    static string dec2bin(int number)
    {
        string binary = String.Empty;
        while (number > 0)
        {
            int reminder = number % 2;
            number /= 2;
            binary += reminder;
        }
        char[] reverse = binary.ToCharArray();
        Array.Reverse(reverse);
        return new string(reverse);
    }

    static string dec2hex(int number)
    {
        string hex = String.Empty;
        while (number > 0)
        {
            int reminder = number % 16;
            number /= 16;
            if (reminder < 10) hex += reminder.ToString();
            else hex += (char)(reminder + 55);
        }
        char[] reverse = hex.ToCharArray();
        Array.Reverse(reverse);
        return new string(reverse);
    }

    static int bin2dec(string bin)
    {
        int result = 0, pow = 0;
        for (int i = bin.Length - 1; i >= 0; i--)
        {
            var A = int.Parse(bin[i].ToString());
            result += (int)(A * Math.Pow(2, pow));
            pow++;
        }
        return result;
    }

    static string bin2hex(string bin)
    {
        string hex = String.Empty;
        while (bin.Length >= 4)
        {
            string sub = bin.Substring(bin.Length - 4, 4);
            switch (sub)
            {
                case "0000": hex += "0"; break;
                case "0001": hex += "1"; break;
                case "0010": hex += "2"; break;
                case "0011": hex += "3"; break;
                case "0100": hex += "4"; break;
```

```
                case "0101": hex += "5"; break;
                case "0110": hex += "6"; break;
                case "0111": hex += "7"; break;
                case "1000": hex += "8"; break;
                case "1001": hex += "9"; break;
                case "1010": hex += "A"; break;
                case "1011": hex += "B"; break;
                case "1100": hex += "C"; break;
                case "1101": hex += "D"; break;
                case "1110": hex += "E"; break;
                case "1111": hex += "F"; break;
            }
            bin = bin.Substring(0, bin.Length - 4);
        }
        switch (bin.Length)
        {
            case 0: bin = "0000"; break;
            case 1: bin = "000" + bin; break;
            case 2: bin = "00" + bin; break;
            case 3: bin = "0" + bin; break;
        }
        switch (bin)
        {
            case "0000": hex += "0"; break;
            case "0001": hex += "1"; break;
            case "0010": hex += "2"; break;
            case "0011": hex += "3"; break;
            case "0100": hex += "4"; break;
            case "0101": hex += "5"; break;
            case "0110": hex += "6"; break;
            case "0111": hex += "7"; break;
            case "1000": hex += "8"; break;
            case "1001": hex += "9"; break;
            case "1010": hex += "A"; break;
            case "1011": hex += "B"; break;
            case "1100": hex += "C"; break;
            case "1101": hex += "D"; break;
            case "1110": hex += "E"; break;
            case "1111": hex += "F"; break;
        }
        char[] reverse = hex.ToCharArray();
        Array.Reverse(reverse);
        return new string(reverse);
    }

    static string hex2bin(string hex)
    {
        string bin = String.Empty;
        for (int i = 0; i < hex.Length; i++)
            switch (hex[i])
            {
                case '0': bin += "0000"; break;
                case '1': bin += "0001"; break;
                case '2': bin += "0010"; break;
                case '3': bin += "0011"; break;
```

```
                case '4': bin += "0100"; break;
                case '5': bin += "0101"; break;
                case '6': bin += "0110"; break;
                case '7': bin += "0111"; break;
                case '8': bin += "1000"; break;
                case '9': bin += "1001"; break;
                case 'A': bin += "1010"; break;
                case 'B': bin += "1011"; break;
                case 'C': bin += "1100"; break;
                case 'D': bin += "1101"; break;
                case 'E': bin += "1110"; break;
                case 'F': bin += "1111"; break;
            }
        return bin;
    }

    static int hex2dec(string hex)
    {
        string bin = hex2bin(hex);
        return bin2dec(bin);
    }

    static void Main(string[] args)
    {
        // (10) -> (2), (10) -> (16)
        Console.WriteLine("{0} (10) = {1} (2)", 1234, dec2bin(1234));
        Console.WriteLine("{0} (10) = {1} (16)", 1234, dec2hex(1234));

        // (2) -> (10), (2) -> (16)
        Console.WriteLine("{0} (2) = {1} (10)", "1100101", bin2dec("1100101"));
        Console.WriteLine("{0} (2) = {1} (16)", "1100101", bin2hex("1100101"));

        // (16) -> (10), (16) -> (2)
        Console.WriteLine("{0} (16) = {1} (10)", "ABC", hex2dec("ABC"));
        Console.WriteLine("{0} (16) = {1} (2)", "ABC", hex2bin("ABC"));
    }
}
}
```

Задача 2.2. P-ична бройна система

Запишете най-малкото и най-голямото двуцифрено число в P-ична бройна система, запишете в същата бройна система на колко е равна тяхната разлика.

Решение

```
namespace PNumeralSystem
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("System = ");
            int P = int.Parse(Console.ReadLine());
            var MIN = P;
            var MAX = Math.Pow(P, 2f) - 1f;
```

```
            var DIFF = MAX - MIN;
            Console.WriteLine("MIN = {0}", MIN);
            Console.WriteLine("MAX = {0}", MAX);
            Console.WriteLine("DIFF = {0}", DIFF);
        }
    }
}
```

Задача 2.3. Числото 111111111111111

Дадено е числото 111111111111111 в двоична бройна система, без да преобразувате числото в десетична бройна система определете неговата четност и запишете числото, което е:

- с 2 по-малко
- с 2 по-голямо
- 2 пъти по-голямо

Решение

```
namespace Number111111111111111
{
    class Program
    {
        static string Minus10(string bin)
        {
            int MIN = 0;
            String MIN10 = bin.Substring(bin.Length - 1, 1);
            for (int i = bin.Length - 2; i >= 0; i--)
            {
                if (i == bin.Length - 2)
                {
                    if (bin[i] == '1')
                    {
                        MIN10 += '0';
                        MIN = 0;
                    }
                    else
                    {
                        MIN10 += '1';
                        MIN = 1;
                    }
                }
                else
                {
                    if (MIN == 0) MIN10 += bin[i];
                    else if (bin[i] == '1')
                    {
                        MIN10 += '0';
                        MIN = 0;
                    }
                    else
                    {
                        MIN10 += '1';
                        MIN = 1;
                    }
                }
```

```
        }

        char[] reverse = MIN10.ToCharArray();
        Array.Reverse(reverse);
        return new string(reverse);
    }

    static string Plus10(string bin)
    {
        int EXTRA = 0;
        String PLUS10 = bin.Substring(bin.Length - 1, 1);
        for (int i = bin.Length - 2; i >= 0; i--)
        {
            if (i == bin.Length - 2)
            {
                if (bin[i] == '1')
                {
                    PLUS10 += '0';
                    EXTRA = 1;
                }
                else
                {
                    PLUS10 += '1';
                    EXTRA = 0;
                }
            }
            else
            {
                if (bin[i] == '1' && EXTRA == 1)
                {
                    PLUS10 += '0';
                    EXTRA = 1;
                }
                else
                {
                    if (bin[i] == '1' || EXTRA == 1) PLUS10 += '1';
                    else PLUS10 += '0';
                    EXTRA = 0;
                }
            }
        }
        if (EXTRA == 1) PLUS10 += '1';

        char[] reverse = PLUS10.ToCharArray();
        Array.Reverse(reverse);
        return new string(reverse);
    }

    static string DoubleIt(string bin)
    {
        int EXTRA = 0;
        String RES = String.Empty;
        for (int i = bin.Length - 1; i >= 0; i--)
        {
            if (i == bin.Length - 1)
```

```
                {
                    RES += '0';
                    if (bin[i] == '1') EXTRA = 1;
                    else EXTRA = 0;
                }
                else
                {
                    RES += EXTRA.ToString();
                    if (bin[i] == '1') EXTRA = 1;
                    else EXTRA = 0;
                }
            }
            if (EXTRA == 1) RES += '1';

            char[] reverse = RES.ToCharArray();
            Array.Reverse(reverse);
            return new string(reverse);
        }

        static void Main(string[] args)
        {
            Console.WriteLine("Enter Binary: ");
            String bin = Console.ReadLine();
            Console.WriteLine("\nMinus 10:\n{0}", Minus10(bin));
            Console.WriteLine("\nPlus 10:\n{0}", Plus10(bin));
            Console.WriteLine("\nDouble:\n{0}", DoubleIt(bin));
        }
    }
}
```

Задача 2.4. Двоична аритметика

Преобразувайте числата в двоична бройна система и извършете действията в двоична бройна система. След това преобразувайте резултата в десетична бройна система

12+15=
9+15=
25-10=
45-17=
13*5=
17*3=
36/4=
81/9=

Решение

```
namespace BinaryAritmetics
{
    internal class Program
    {
        static string dec2bin(int number)
        {
            string binary = String.Empty;
            while (number > 0)
```

```
            {
                int reminder = number % 2;
                number /= 2;
                binary += reminder;
            }
            char[] reverse = binary.ToCharArray();
            Array.Reverse(reverse);
            return new string(reverse);
        }

        static void Main(string[] args)
        {
            Console.WriteLine($"12+15={dec2bin(12)}+{dec2bin(15)}={12 + 15}");
            Console.WriteLine($"9+15={dec2bin(9)}+{dec2bin(15)}={9 + 15}");
            Console.WriteLine($"25-10={dec2bin(25)}-{dec2bin(10)}={25 - 10}");
            Console.WriteLine($"45-17={dec2bin(45)}-{dec2bin(17)}={45 - 17}");
            Console.WriteLine($"13*5={dec2bin(13)}*{dec2bin(5)}={13 * 5}");
            Console.WriteLine($"17*3={dec2bin(17)}*{dec2bin(3)}={17 * 3}");
            Console.WriteLine($"36/4={dec2bin(36)}/{dec2bin(4)}={36 / 4}");
            Console.WriteLine($"81/9={dec2bin(81)}/{dec2bin(9)}={81 / 9}");
        }
    }
}
```

Задача 2.5. Шестнадесетична аритметика

Преобразувайте числата в шестнадесетична бройна система и извършете действията в шестнадесетична бройна система. След това преобразувайте резултата в десетична бройна система

12+15=
9+15=
25-10=
45-17=
13*5=
17*3=
36/4=
81/9=

Решение

```
namespace HexadecimalAritmetics
{
    internal class Program
    {
        static string dec2hex(int number)
        {
            string hex = String.Empty;
            while (number > 0)
            {
                int reminder = number % 16;
                number /= 16;
                if (reminder < 10) hex += reminder.ToString();
                else hex += (char)(reminder + 55);
            }
```

```
            char[] reverse = hex.ToCharArray();
            Array.Reverse(reverse);
            return new string(reverse);
        }

        static void Main(string[] args)
        {
            Console.WriteLine($"12+15={dec2hex(12)}+{dec2hex(15)}={12 + 15}");
            Console.WriteLine($"9+15={dec2hex(9)}+{dec2hex(15)}={9 + 15}");
            Console.WriteLine($"25-10={dec2hex(25)}-{dec2hex(10)}={25 - 10}");
            Console.WriteLine($"45-17={dec2hex(45)}-{dec2hex(17)}={45 - 17}");
            Console.WriteLine($"13*5={dec2hex(13)}*{dec2hex(5)}={13 * 5}");
            Console.WriteLine($"17*3={dec2hex(17)}*{dec2hex(3)}={17 * 3}");
            Console.WriteLine($"36/4={dec2hex(36)}/{dec2hex(4)}={36 / 4}");
            Console.WriteLine($"81/9={dec2hex(81)}/{dec2hex(9)}={81 / 9}");
        }
    }
}
```

Задача 2.6. Векове към минути

Напишете програма, която въвежда цяло число - брой векове и преобразува към години, дни, часове и минути.

Примери

Вход	Изход
1	1 centuries = 100 years = 36524 days = 876576 hours = 52594560 minutes
5	5 centuries = 500 years = 182621 days = 4382904 hours = 262974240 minutes

Подсказки

- Използвайте подходящ тип данни, за да се събере всяко преобразуване
- Нека годината има 365.2422 дни (Тропическа година).

Решение

```
namespace CenturiesToMinutes
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter centuries:");
            int vek = int.Parse(Console.ReadLine());
            int god = vek * 100;
            int dni = (int)(god * 365.2422);
            int chas = 24 * dni;
            int min = chas * 60;
            Console.WriteLine("{0} centuries = {1} years = {2} days = {3} hours =
{4} minutes", vek, god, dni, chas, min);
        }
    }
}
```

Задача 2.7. Цели числа

Напишете програма, която присвоява цели стойности на променливи. Уверете се, че всяка стойност е записана в правилния тип (във всеки случай използвайте възможно най-икономичния тип по отношение на паметта). Накрая изведете всички променливи в конзолата

Примери

Вход	Изход
-100 128 -3540 64876 2147483648 -1141583228 -1223372036854775808	-100 128 -3540 64876 2147483648 -1141583228 -1223372036854775808

Решение

```
namespace IntegerNumbers
{
    class Program
    {
        static void Main(string[] args)
        {
            sbyte num1 = -100;
            byte num2 = 128;
            short num3 = -3540;
            ushort num4 = 64876;
            uint num5 = 2147483648;
            int num6 = -1141583228;
            long num7 = -1223372036854775808;

            Console.WriteLine(num1);
            Console.WriteLine(num2);
            Console.WriteLine(num3);
            Console.WriteLine(num4);
            Console.WriteLine(num5);
            Console.WriteLine(num6);
            Console.WriteLine(num7);
        }
    }
}
```

Задача 2.8. Шестнадесетична променлива

Напишете програма, която въвежда стойност в шестнадесетичен формат (0x##) и я преобразува в десетичен формат, след което извежда стойността.

Примери

Вход	Изход
0xFE	254

Вход	Изход
0x37	55

Вход	Изход
0x10	16

Подсказки

- Използвайте: Convert.ToInt32(string, 16).

Решение

```
namespace HexadecimalVariable
{
    class Program
    {
        static void Main(string[] args)
        {
            string a = Console.ReadLine();
            Console.WriteLine(Convert.ToInt32(a, 16));
        }
    }
}
```

Задача 2.9. Размяна на стойности на променливи

Декларирайте две целочислени променливи a и b и им присвоете стойности 5 и 10 след това разменете техните стойности чрез някаква програмна логика. Изведете стойностите на променливите преди и след размяната, както е показано:

Примери

Вход	Изход
5 10	Before: a = 5 b = 10 After: a = 10 b = 5

Подсказки

Трябва да използвате временна променлива, за да запомните старата стойност на a, след което запишете стойността на b в a, тогава запишете в b стойността на времената променлива.

Решение

```
namespace ExchangeVariablesValues
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = int.Parse(Console.ReadLine());
            int b = int.Parse(Console.ReadLine());
```

```
            var c = 0;

            Console.WriteLine("Before");
            Console.WriteLine(a);
            Console.WriteLine(b);

            c = a;
            a = b;
            b = c;

            Console.WriteLine("After");
            Console.WriteLine(a);
            Console.WriteLine(b);
        }
    }
}
```

Задача 2.10. Десетично към шестнадесетично и двоично

Напишете програма, която преобразува десетично число в шестнадесетично и двоично число и го извежда.

Примери:

Вход	Изход
10	A 1010

Вход	Изход
420	1A4 110100100

Вход	Изход
256	100 100000000

Подсказки:

- Използвайте: Convert.ToString(number, base) и string.ToUpper().

Решение

```
namespace DecimalToHexadecimal
{
    class Program
    {
        static void Main(string[] args)
        {
            var a = int.Parse(Console.ReadLine());
            string b = Convert.ToString(a, 16).ToUpper();
            string c = Convert.ToString(a, 2).ToUpper();
            Console.WriteLine(b);
            Console.WriteLine(c);
        }
    }
}
```

Задача 2.11. Делене на цели числа

Напишете програма, която въвежда едно цяло число n. След това програмата въвежда 2n на брой цели числа, всяко на отделен ред. Програмата да извежда целочислената загуба от деленията на всяка двойка числа. Целочислена загуба дефинираме като сумата от остатъците от деленето на: първото на второто число, третото на четвъртото число и т.н.

Примери:

Вход	Изход	Обяснение
5 1 2 5 2 4 2 10 5 8 3	4	1 / 2 = 0 и ост. 1 5 / 2 = 2 и ост. 1 4 / 2 = 2 и ост. 0 10 / 5 = 2 и ост. 0 8 / 3 = 1 и ост. 2 Събираме всички остатъци и получаваме: 1 + 1 + 0 + 0 + 2 = 4

Решение

```
namespace DivisionOfIntegers
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            int a, b, sum = 0;
            for (int i = 0; i < n; i++)
            {
                a = int.Parse(Console.ReadLine());
                b = int.Parse(Console.ReadLine());
                sum += a % b;
            }
            Console.WriteLine(sum);
        }
    }
}
```

Задача 2.12. Числа с плаваща запетая

Напишете програма, която присвоява числа с плаваща запетая на променливи. Уверете се, че всяка стойност се запазва в коректен тип (изберете най-удобния тип спрямо количеството паметта, което той заема). Накрая трябва да изведете всички променливи.

Примери

Вход	Изход
3.141592653589793238 1.60217657 7.8184261974584555216535342341	3.141592653589793238 1.60217657 7.8184261974584555216535342341

Решение

```
namespace FloatingPointNumbers
{
    class Program
    {
        static void Main(string[] args)
        {
            decimal a = 3.141592653589793238m;
            double b = 1.60217657d;
            decimal c = 7.818426197458455521653535342341m;
            Console.WriteLine(a);
            Console.WriteLine(b);
            Console.WriteLine(c);
        }
    }
}
```

Задача 2.13. Лице на кръг (с точност 12 знака)

Напишете програма, в която въвеждаме радиус r (реално число) и извеждаме лицето на кръг с точно 12 знака след десетичната запетая. Използвайте тип данни с подходяща точност за съхранение на резултатите.

Пример

Вход	Изход
2.5	19.634954084936

Вход	Изход
1.2	4.523893421169

Подсказки

- Може да използвате тип **double**. Той има точност 15-16 знака.
- За да изведете точно 12 знака след десетичната запетая, може да ползвате следния код:

```
double r = double.Parse(Console.ReadLine());
Console.WriteLine("{0:f12}", Math.PI * r * r);
```

Решение

```
namespace CircleArea
{
    class Program
    {
        static void Main(string[] args)
        {
            var r = double.Parse(Console.ReadLine());
            var area = Math.PI * r * r;
            Console.WriteLine("{0:f12}", area);
        }
    }
}
```

Задача 2.14. Точна сума на реални числа

Напишете програма, която въвежда n числа и изчислява и извежда тяхната точна сума (без закръгляне).

Примери

Вход	Изход
3 1000000000000000000 5 10	1000000000000000015

Вход	Изход
2 0.00000000003 333333333333.3	333333333333.30000000003

Подсказки

- Ако използвате типове като float или double, резултатът ще изгуби точността си. Също така данните може да се изведат с експоненциален запис. Може да използвате типа decimal, който съхранява реални числа с висока точност и по-малка загуба.
- Забележете, че decimal понякога съдържа ненужните нули след десетичната запетая, т.е. 0m е различно спрямо 0.0m и 0.00000m.

Решение

```
namespace DecimalNumbersSum
{
    class Program
    {
        static void Main(string[] args)
        {
            var n1 = decimal.Parse(Console.ReadLine());
            var n2 = decimal.Parse(Console.ReadLine());
            var n3 = decimal.Parse(Console.ReadLine());
            var n4 = decimal.Parse(Console.ReadLine());
            Console.WriteLine(n1 + n2 + n3 + n4);
        }
    }
}
```

Задача 2.15. Правоъгълник

Напишете програма, която изчислява за даден правоъгълник неговите обиколка, лице и диагонал по неговите страни.

Примери

Вход	Изход
10 5	30 50 11.1803398874989

Вход	Изход
22.1 10.2	64.6 225.42 24.3402958075698

Подсказки

- Използвайте `Math.Sqrt()` за да изчислите диагонала (използвайте $c^2 = a^2 + b^2$). Разгледайте http://www.mathopenref.com/rectangulardiagonals.html.

Решение

```
namespace RectangleCalculations
{
    class Program
    {
        static void Main(string[] args)
        {
            double a = double.Parse(Console.ReadLine());
            double b = double.Parse(Console.ReadLine());
            double P = 2 * a + 2 * b;
            double S = a * b;
            decimal d = (decimal)Math.Sqrt(Math.Pow(a, 2) + Math.Pow(b, 2));
            Console.WriteLine(P);
            Console.WriteLine(S);
            Console.WriteLine(d);
        }
    }
}
```

Задача 2.16. Преобразуване на скорост

Напишете програма, която въвежда разстояние (в метри) и време (като три числа: часове, минути, секунди), и изведете скоростта, в метри за секунда, километри в час и мили в час.

Приемете, че 1 миля = 1609 метра.

Вход

- На първите ред ще получите – разстояние в метри
- На втория – часове
- На третия – минути
- На четвъртия – секунди

Изход

Всяко число в изхода трябва да бъде изведено с точност 6 знака след запетаята:

- На първи ред – скоростта в метри в секунди (m/s)
- На втори ред – скоростта в километри в час (km/h)
- На трети ред – скоростта в мили в час (mp/h)

Примери

Вход	Изход
1000 1 1 0	0.2732241 0.9836066 0.6113155

Вход	Изход
10000 0 20 30	8.130081 29.26829 18.19036

Вход	Изход
200000 2 5 0	26.66667 96 59.66439

Подсказки

- Потърсете в интернет как да преобразувате мерните единици за скорост

- Типът float е достатъчно голям за тези изчисления.

Решение

```
namespace SpeedConversion
{
    class Program
    {
        static void Main(string[] args)
        {
            var n = double.Parse(Console.ReadLine());
            var h = double.Parse(Console.ReadLine());
            var m = double.Parse(Console.ReadLine());
            var s = double.Parse(Console.ReadLine());
            var t = h * 3600 + m * 60 + s;
            var ms = n / t;
            var kmh = (n / 1000) / (t / 3600);
            var mp = (n / 1609) / (t / 3600);
            Console.WriteLine("{0:f6}", ms);
            Console.WriteLine("{0:f6}", kmh);
            Console.WriteLine("{0:f6}", mp);
        }
    }
}
```

Задача 2.17. Асансьор

Изчислете колко курса ще трябва да направя един асансьор, за да се качат n човека, ако капацитета на асансьора е p човека. Входа се състои от два реда: броя на хората n и капацитета p на асансьора.

Примери

Вход	Изход	Коментар
17 3	6	5 курса * 3 човека + 1 курс * 2 човека
4 5	1	Всички хора се побират в асансьора. Един курс е достатъчен.
10 5	2	2 курса * 5 човека

Подсказки

- Трябва да разделите **n** на **p**. Това дава броя на курсове с пълен капацитет (e.g. 17 / 3 = 5).
- Ако **n** не се дели точно на **p**, то ще трябва да се направи още един частично пълен курс (например 17 % 3 = 2).
- Друг начин е да се закръгли нагоре **n / p** към най-близкото цяло число (използвайки Math.Ceiling), например 17/3 = 5.67 → се закръгля нагоре до 6.
- Примерен код за закръгляне:

```
int courses = (int)Math.Ceiling((double)n / p);
```

Решение

```
namespace Lift
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            int p = int.Parse(Console.ReadLine());
            int courses = (int)Math.Ceiling((double)n / p);
            Console.WriteLine(courses);
        }
    }
}
```

Задача 2.18. Специални числа

Едно число наричаме специално когато неговата сума от цифри е 5, 7 или 11.

Напишете програма, която въвежда цяло число n и за всички числа в интервала 1...n извежда дали числото е специално или не (True / False).

Примери

Вход	Изход
15	1 -> False 2 -> False 3 -> False 4 -> False 5 -> True 6 -> False 7 -> True 8 -> False 9 -> False 10 -> False 11 -> False 12 -> False 13 -> False 14 -> True 15 -> False

Подсказки

Изчислете сумата на цифрите на даденото число num, повтаряйте следната процедура: добавете към сума последната цифра (num % 10) и я премахнете от записа (използвайте целочислено делене num = num / 10) докато num стигне 0.

Решение

```
namespace SpecialNumbers
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            for (int num = 1; num <= n; num++)
            {
                int sumOfDigits = 0;
                int digits = num;
                while (digits > 0)
                {
                    sumOfDigits += digits % 10;
                    digits = digits / 10;
                }
                bool special = (sumOfDigits == 5) || (sumOfDigits == 7) ||
(sumOfDigits == 11);
                Console.WriteLine("{0} -> {1}", num, special);
            }

        }
    }
}
```

Задача 2.19. Булева променлива

Напишете програма, която въвежда низ, преобразува го към променлива от булев тип и извежда "Yes" ако в променливата имаме true и "No" ако в променливата имаме false.

Примери

Вход	Изход
True	Yes
False	No

Подсказки

- Използвайте: Convert.ToBoolean(string).

Решение

```
namespace BooleanVariable
{
    class Program
    {
        static void Main(string[] args)
        {
            string n = Console.ReadLine();
            Convert.ToBoolean(n);
            if (n == "True") Console.WriteLine("Yes");
            else Console.WriteLine("No");
        }
    }
}
```

Задача 2.20 Тройки латински букви

Напишете програма, която въвежда цяло число n и отпечатва всички тройки от първите n малки латински букви. Използвайте азбучна подредба

Примери

Вход	Изход
3	aaa aab aac aba abb abc aca acb acc baa bab bac bba bbb bbc bca bcb bcc caa cab cac cba cbb cbc cca ccb ccc

Подсказки

Изпълнете три вложени цикъла от **0** до **n-1**. За всяко число **num** изведете съответната му латинска буква:

```
char letter = (char)('a' + num);
```

Решение

```
namespace ThreeLatinletters
{
    class Program
```

```
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            for (int i1 = 0; i1 < n; i1++)
                for (int i2 = 0; i2 < n; i2++)
                    for (int i3 = 0; i3 < n; i3++)
                    {
                        char letter1 = (char)('a' + i1);
                        char letter2 = (char)('a' + i2);
                        char letter3 = (char)('a' + i3);
                        Console.WriteLine("{0}{1}{2}",
                            letter1, letter2, letter3);
                    }
        }
    }
}
```

Задача 2.21. Поздрав

Напишете програма, която въвежда първото име, последното име и възрастта и извежда "*Hello, <first name> <last name>. You are <age> years old.*". Използвайте съставни низове.

Примери

Вход	Изход
Ivo Hristov 23	Hello, Ivo Hristov. You are 23 years old.

Подсказки

Може да използвате подобен код:

```
Console.WriteLine(
    $"Hello, {firstName} {lastName}.\r\nYou are {age} years old.");
```

Решение

```
namespace Greetings
{
    class Program
    {
        static void Main(string[] args)
        {
            string name = Console.ReadLine();
            string surName = Console.ReadLine();
            int age = int.Parse(Console.ReadLine());
            object hello = name + " " + surName;
            Console.WriteLine("Hello {0}. You are {1} years old.", hello, age);
        }
    }
```

```
}
```

Задача 2.22. Низове и знаци

Напишете програма, която задава знакови и низови стойности на променливи. Бъдете сигурни, че всяка стойност се съхранява в коректната променлива. Накрая, трябва да отпечатаме всички променливи на конзолата.

Примери

Вход	Изход
A string is sequence of chars B y e I love programming	A string is sequence of chars B y e I love programming

Подсказки

Декларирайте променливи от тип char или string, задайте им съответните стойности и ги отпечатайте.

Решение

```
namespace StringsAndCharacters
{
    class Program
    {
        static void Main(string[] args)
        {
            string first = Console.ReadLine();
            string second = Console.ReadLine();
            string third = Console.ReadLine();
            string fourth = Console.ReadLine();
            string fifth = Console.ReadLine();
            Console.WriteLine(first);
            Console.WriteLine(second);
            Console.WriteLine(third);
            Console.WriteLine(fourth);
            Console.WriteLine(fifth);
        }
    }
}
```

Задача 2.23. Низове и обекти

Декларирайте две променливи от тип string и им задайте стойности "Hello" и "World". Декларирайте променлива от тип object и ѝ присвоете слепването на първите две променливи (не забравяйте да добавите интервал между тях). Направете трета променлива от тип string и я инициализирайте със стойността от променливата с тип object (трябва да извършите преобразуване).

Примери

Вход	Изход
Hello World	Hello World

Решение

```
namespace StringAndObjects
{
    class Program
    {
        static void Main(string[] args)
        {
            string lineI = Console.ReadLine();
            string lineII = Console.ReadLine();
            object merge = lineI + " " + lineII;
            Console.WriteLine(merge);
        }
    }
}
```

Задача 2.24. Обръщане на знаци

Напишете програма, която въвежда 3 знака и ги изведете в обратен ред.

Примери

Вход	Изход
A B C	CBA

Вход	Изход
x Y z	zYx

Вход	Изход
G g n	ngG

Решение

```
namespace CharacterReverse
{
    class Program
    {
        static void Main(string[] args)
        {
            string symbol1 = Console.ReadLine();
            string symbol2 = Console.ReadLine();
            string symbol3 = Console.ReadLine();
            Console.Write($"{symbol3}{symbol2}{symbol1}");
        }
    }
}
```

Задача 2.25. Данни на служител

Маркетинг компания иска да пази информация за служителите си. Всеки запис съдържа следната информация:

- Име
- Фамилия

- Възраст (0...100)
- Пол (`m` или `f`)
- ЕГН (e.g. 8306112507)
- Уникален номер на служителя (27560000...27569999)

Декларирайте променливите необходими да се пази информацията за един служител използвайки подходящи типове данни. Използвайте описателни имена. Изведете данните на конзолата.

Примери

Вход	Изход
Amanda Jonson 27 f 8306112507 27563571	First name: Amanda Last name: Jonson Age: 27 Gender: f Personal ID: 8306112507 Unique Employee number: 27563571

Подсказки

```
string firstName = "Amanda";
// TODO ...
int employeeNumber = 27563571;

Console.WriteLine(firstName);
// TODO ...
Console.WriteLine(employeeNumber);
```

Решение

```
namespace EmployeeData
{
    class Program
    {
        static void Main(string[] args)
        {
            string firstName = Console.ReadLine();
            string lastName = Console.ReadLine();
            string Years = Console.ReadLine();
            string gender = Console.ReadLine();
            string personalID = Console.ReadLine();
            string employeeNumber = Console.ReadLine();

            Console.WriteLine($"First name : {firstName}");
            Console.WriteLine($"Last name : {lastName}");
            Console.WriteLine($"Age : {Years}");
            if (gender == "f")
            {
                Console.WriteLine($"Gender : f ");
            }
```

```
            else if (gender == "m")
            {
                Console.WriteLine($"Gender : m ");
            }
            Console.WriteLine($"Personal ID : {personalID}");
            Console.WriteLine($"Unique Employee number : {employeeNumber}");
        }
    }
}
```

Задача 2.26. Рефакторирайте Обем на пирамида

Получавате работещ код, който намира обема на пирамида. Въпреки това, трябва да вземете предвид неговото качество – дали променливите са именувани разумно, дали се използват най-подходящите типове, какъв е техния промеждутък и дали се използват само за едно действие.

Код

Примерен код

```
double dul, sh, V = 0;
Console.Write("Length: ");
dul = double.Parse(Console.ReadLine());
Console.Write("Width: ");
sh = double.Parse(Console.ReadLine());
Console.Write("Heigth: ");
V = double.Parse(Console.ReadLine());
V = (dul + sh + V) / 3;
Console.WriteLine("Pyramid Volume: {0:F2}", V);
```

Подсказки

- Намалете промеждутъка на променливите като ги декларирате в момента, в който те получат стойности, а не преди това
- Преименувайте променливите, така че да показват тяхното истинско предназначение (например: "dul" трябва да стане дължина, и т.н.)
- Проверете за променливи, които се използват с няколко предназначения. Създайте нови променливи.

Решение

```
namespace PyramidVolume
{
    class Program
    {
        static void Main(string[] args)
        {
            var Lenght = double.Parse(Console.ReadLine());
            var Width = double.Parse(Console.ReadLine());
            var Height = double.Parse(Console.ReadLine());
            var V = (Lenght + Width + Height) / 3;
            Console.Write("Length: {0} ", Lenght);
```

```
            Console.Write("Width: {0} ", Width);
            Console.Write("Heigth: {0} ", Height);
            Console.WriteLine("Pyramid Volume: {0:F2}", V);
        }
    }
}
```

Задача 2.27. Граници на типа

Напишете програма, която получава числен тип (като низ) и отпечатва максималната и минималната стойност на съответния тип. Ще получите един от следните типове: int, uint, long, byte и sbyte.

Примери

Вход	Изход
int	2147483647 -2147483648

Вход	Изход
byte	255 0

Подсказки

Следвайте идеята от този код:

```
switch (type)
{
    case "int":
        Console.WriteLine(int.MaxValue);
        Console.WriteLine(int.MinValue);
        break;
   // Add the other cases
    case "sbyte":
        Console.WriteLine(sbyte.MaxValue);
        Console.WriteLine(sbyte.MinValue);
        break;
}
```

Решение

```
namespace TypeBoundaries
{
    internal class Program
    {
        static void Main(string[] args)
        {
            string type = Console.ReadLine();
            switch (type)
            {
                case "int":
                    Console.WriteLine(int.MaxValue);
                    Console.WriteLine(int.MinValue);
                    break;

                case "uint":
                    Console.WriteLine(uint.MaxValue);
```

```
                    Console.WriteLine(uint.MinValue);
                    break;

                case "long":
                    Console.WriteLine(long.MaxValue);
                    Console.WriteLine(long.MinValue);
                    break;

                case "byte":
                    Console.WriteLine(byte.MaxValue);
                    Console.WriteLine(byte.MinValue);
                    break;

                case "sbyte":
                    Console.WriteLine(sbyte.MaxValue);
                    Console.WriteLine(sbyte.MinValue);
                    break;

                default:
                    Console.WriteLine("Supported types: int, uint, long, byte и
sbyte.");
                    break;
            }
        }
    }
}
```

Задача 2.28. Проверка на число

Напишете програма, която проверява дали дадено число е цяло или с плаваща запетая и изведете "floating-point" или "integer", според случая. Ще бъдат въвеждани само числа.

Ограничения

Целите числа ще са в интервала [-9223372036854775808...9223372036854775807]

Примери

Вход	Изход
3	integer

Вход	Изход
2.31	floating-point

Решение

```
namespace NumberCheck
{
    internal class Program
    {
        static void Main(string[] args)
        {
            var value = Console.ReadLine();

            float f;
            int i;

            if (int.TryParse(value, out i))
```

```
            {
                Console.WriteLine("integer");
            }
            else if (float.TryParse(value, out f))
            {
                Console.WriteLine("floating-point");
            }
        }
    }
}
```

Задача 2.29. Преливане на вода

Имате съд за вода с капацитет от 255 литра. На следващите n реда, ще получите литри вода, които трябва да налеете във вашия съд. Ако капацитета на вашия съд не е достатъчен, изведете Insufficient capacity! и продължете със следващия ред. На последния ред, изведете литрите в съда.

Вход

Входът ще се състои от 2 реда:

- На първи ред, ще получите n – брой редове, които ще следват
- От следващите n реда – ще получите количествата вода, които ще трябва да наливате в съда

Изход

Всеки път когато нямате достатъчно капацитет в съда, извеждайте:

Insufficient capacity!

На последния ред, изведете само литрите в съда.

Ограничения

- n ще е в интервала [1...20]
- liters ще е в интервала [1...1000]

Примери

Вход	Изход
5 20 100 100 100 20	Insufficient capacity! 240

Вход	Изход
1 1000	Insufficient capacity! 0

Вход	Изход
7	105

Вход	Изход
4	Insufficient capacity!

10 20 30 10 5 10 20		250 10 20 40	Insufficient capacity! Insufficient capacity! 250

Решение

```
namespace WaterOverflow
{
    public static class Program
    {

        static void Main(string[] args)
        {
            int capacity = 255;
            int n = int.Parse(Console.ReadLine());
            for (int i = 0; i < n; i++)
            {
                int next = int.Parse(Console.ReadLine());
                if (capacity - next < 0)
                {
                    Console.WriteLine("Insufficient capacity!");
                    continue;
                }
                capacity -= next;
            }
            Console.WriteLine(255 - capacity);
        }
    }
}
```

Задача 2.30 Туристическа информация

Напишете програма, която помага на туристите да преобразуват имперални мерки към метричната система. Вашата програма трябва да поддържа следните преобразувания: мили към километри, инчове към сантиметри, футове към сантиметри, ярдове към метри и галони към литри. Таблицата за преобразуване е:

Имаме:	Умножаваме по:	Получаваме
miles	1.6	kilometers
inches	2.54	centimeters
feet	30	centimeters
yards	0.91	meters
gallons	3.8	liters

Вход

Входът се състои от два реда:

- На първи ред, ще получите мярка от имперската система, която трябва да преобразувате
- На втори ред, ще получите стойността, която трябва да преобразувате

Изход

Изведете отговора в следния формат:

{начална стойност} {начална мярка} = {преобразувана стойност} {преобразувана мярка }

Форматирайте преобразувана стойност до 2ри знак след запетаята.

Изведете началната стойност така както е дадена.

Ограничения

- Стойността, която трябва да бъде преобразувана ще бъде в интервала [±1.5×10-45... ±3.4×1038].

Примери

Вход	Изход
miles 12.313	12.313 miles = 19.70 kilometers

Вход	Изход
gallons 12	12 gallons = 45.60 liters

Решение

```
namespace TouristInformation
{
    public static class Program
    {

        static void Main(string[] args)
        {
            string unit = Console.ReadLine();
            int value = int.Parse(Console.ReadLine());
            switch (unit)
            {
                case "miles":
                    Console.WriteLine("{0} {1} = {3:f2} kilometers", value, unit,
value * 1.6);
                    break;

                case "inches":
                    Console.WriteLine("{0} {1} = {3:f2} centimeters", value, unit,
value * 2.54);
                    break;

                case "feet":
                    Console.WriteLine("{0} {1} = {3:f2} centimeters", value, unit,
value * 30);
                    break;
```

```
                case "yards":
                    Console.WriteLine("{0} {1} = {3:f2} meters", value, unit, value
* 0.91);
                    break;

                case "gallons":
                    Console.WriteLine("{0} {1} = {3:f2} litres", value, unit, value
* 3.8);
                    break;
            }
        }
    }
}
```

Задача 2.31. Прогноза за времето

Изобретили сте нова технология за прогнозиране на времето, чрез нумерология. Ще получите число, чрез което може да прогнозирате времето утре. Системата работи по следния начин:

- Ако числото се побира в sbyte – времето е "Sunny"
- Ако числото се побира в int – времето е "Cloudy"
- Ако числото се побира в long – времето е "Windy"
- Ако числото е с плаваща запетая – времето е "Rainy"

Винаги извеждайте най-малкия възможен вариант.

Вход

На първи ред, ще получите число.

Изход

Изведете вашата прогноза за времето.

Ограничения

Всяко цяло число ще бъде в интервала [-9223372036854775808...9223372036854775807].

Примери

Вход	Изход
120	Sunny

Вход	Изход
-1.31	Rainy

Решение

```
namespace WeatherForecast
{
    public static class Program
    {

        static void Main(string[] args)
        {
            var value = long.Parse(Console.ReadLine());
```

```
            if (value is sbyte)
            {
                Console.WriteLine("Sunny");
            }

            if (value is int)
            {
                Console.WriteLine("Cloudy");
            }

            if (value is long)
            {
                Console.WriteLine("Windy");
            }

            if (value is double)
            {
                Console.WriteLine("Rainy");
            }
        }
    }
}
```

Тема 3. Масиви и списъци

Задача 3.1. Статистика на масив

Напишете програма, която получава масив от цели числа (разделени с интервал) и извежда най-малкия елемент, най-големия елемент, сумата на елементите и средната им стойност.

Примери

Вход	Изход
2 3 4 5 6 1	Min = 1 Max = 6 Sum = 21 Average = 3.5

Вход	Изход
-1 200 124123 -400 - 124214	Min = -124214 Max = 124123 Sum = -292 Average = -58.4

Решение

```
namespace ArrayStatistics
{
    class Program
    {
        static void Main(string[] args)
        {
            var n = int.Parse(Console.ReadLine());
            double min = 1000, max = -1000, sum = 0;
            int[] arr = new int[n];
            for (int i = 0; i < n; i++) arr[i] = int.Parse(Console.ReadLine());
            for (int i = 0; i < n; i++)
            {
                sum = sum + arr[i];
```

```
                if (arr[i] < min) min = arr[i];
                if (arr[i] > max) max = arr[i];
            }
            Console.WriteLine(min);
            Console.WriteLine(max);
            Console.WriteLine(sum);
            Console.WriteLine(sum / n);
        }
    }
}
```

Задача 3.2. Най-често срещано число

Напишете програма, която намира най-често срещаното число в дадена последователност.

- Числата ще са в интервала [0...65535].
- В случай, че има няколко най-често срещани числа, изведете най-лявото от тях.

Примери

Вход	Изход	Коментари
4 1 1 4 2 3 4 4 1 2 4 9 3	4	Числото 4 е най-често срещаното (среща се 5 пъти)
2 2 2 2 1 2 2 2	2	Числото 2 е най-често срещаното (среща се 7 пъти)
7 7 7 0 2 2 2 0 10 10 10	7	Числата 2, 7 и 10 имат максимална честота (всяко се среща 3 пъти). Най-лявото е 7.

Решение

```
namespace MostCommonNumber
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = 0;
            int[] num = new int[100];
            var line = Console.ReadLine();
            var splitted = line.Split(' ');

            foreach (var item in splitted)
            {
                num[n++] = int.Parse(item);
            }

            int[] counter = new int[65535];

            for (int i = 0; i < n; i++)
            {
                counter[num[i]]++;
            }
```

```
            int max = -1000, maxi = 0;
            for (int j = 0; j < 65535; j++)
            {
                if (counter[j] > max)
                {
                    max = counter[j];
                    maxi = j;
                }
            }

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

Задача 3.3. Индекс на буква

Напишете програма, която създава масив, съдържащ всички букви от английската азбука (a-z). Въведете дума с малки букви (lowercase) от конзолата и изведете съответния индекс на всяка буква от масива с буквите от английската азбука.

Примери

Вход	Изход
abcz	a -> 0 b -> 1 c -> 2 z -> 25
easter	e -> 4 a -> 0 s -> 18 t -> 19 e -> 4 r -> 17

Решение

```
namespace LetterIndex
{
    internal class Program
    {
        static void Main(string[] args)
        {
            var letters = Console.ReadLine();
            foreach (var letter in letters)
            {
                var index = (int)letter - 97;
                Console.WriteLine($"{letter} -> {index}");
            }
        }
    }
}
```

Задача 3.4. Преобразуване на масив в число

Напишете програма, която въвежда масив от цели числа и г пребразува чрез сумиране на съседни двойки елементи, докато се получи едно цяло число. Например, ако имаме 3 елемента {2,10,3}, то събираме първите два и вторите два елемента и получаваме {2+10, 10+3} = {12, 13}, после събираме всички съседни елементи и получаваме obtain {12+13} = {25}.

Примери

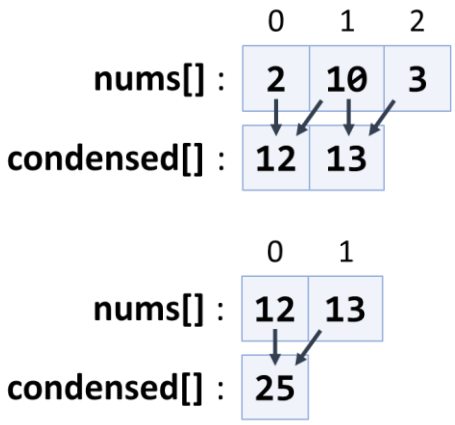
Вход	Изход	Коментари
2 10 3	25	2 10 3 → 2+10 10+3 → 12 13 → 12 + 13 → 25
5 0 4 1 2	35	5 0 4 1 2 → 5+0 0+4 4+1 1+2 → 5 4 5 3 → 5+4 4+5 5+3 → 9 9 8 → 9+9 9+8 → 18 17 → 18+17 → 35
1	1	1 is already condensed to number

Упътване

Докато имаме повече от един елемент в масива nums[], повтаряй следното:

- Създай нов масив condensed[] с размер nums.Length-1.
- Събирай числата от nums[] в condensed[]:
- condensed[i] = nums[i] + nums[i+1]
- nums[] = condensed[]

Процесът е илюстриран по-долу:



Решение

```
namespace ArrayToNumber
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int[] nums = Console.ReadLine().Split(' ').Select(int.Parse).ToArray();
            int[] condensed = new int[nums.Length - 1];
```

```
            int n = nums.Length;
            do
            {
                for (int i = 0; i < n - 1; i++)
                    condensed[i] = nums[i] + nums[i + 1];
                nums = condensed;
                n--;
            } while (n != 1);

            Console.WriteLine(condensed[n - 1]);
        }
    }
}
```

Задача 3.5. Обръщане на последователността на елементите на масив

Напишете програма, която въвежда масив от цели числа, Обръща го и извежда елементите. Входните данни са числото n (брой на елементите) + n цели числа, всяко на отделен ред. Изведете резултата на един ред, за разделител да се ползва интервал

Примери

Вход	Изход
3 10 20 30	30 20 10
4 -1 20 99 5	5 99 20 -1

Упътване

- Първо, въведете числото n.
- Създайте масив от n цели числа.
- Въведете с цикъл for числата.
- Вместо да обръщате масива, можете просто да изведете елементите му като го обходите от последния до първия

Решение

```
namespace ArrayReverse
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
            int n = int.Parse(Console.ReadLine());
            int[] nums = new int[n];

            for (int i = 0; i < n; i++)
            {
                nums[i] = int.Parse(Console.ReadLine());
            }

            for (int i = n - 1; i >= 0; i--)
            {
                Console.Write("{0} ", nums[i]);
            }
        }
    }
}
```

Задача 3.6. Обръщане на масив от символни низове

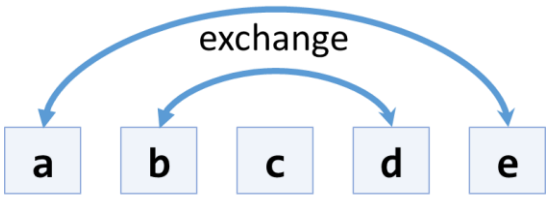
Напишете програма, която да прочете масив от символни низове, обръща масива и печата на неговите елементи. Входът се състои от поредица от низове, разделени с интервал. Отпечатва резултата на един ред с разделител интервал.

Примери

Вход	Изход
a b c d e	e d c b a
-1 hi ho w	w ho hi -1

Упътване

- Въведете масив от символни низове
- Разменете първият елемент (с индекс 0) с последния елемент (с индекс n--1)
- Продължете с тези размени с останалите елементи докато стигнете средата на масива



- Друг, по-кратък подход е да се ползва готовия extension метод .Reverse() от System.Linq.

Решение

```
namespace StringReverse
{
    class Program
    {
        static void Main(string[] args)
```

```
        {
            var arr = Console.ReadLine().Split(' ');
            for (int i = arr.Length - 1; i >= 0; i--)
            {
                Console.Write("{0} ", arr[i]);
            }
        }
    }
}
```

Задача 3.7. Завъртане и сумиране

"Завъртане на масив на дясно" означава да преместим неговия последен елемент на първо място: {1, 2, 3} → {3, 1, 2}.

Напишете програма, която въвежда масив от n цели числа (разделени с интервал на един ред) и цяло число k, завърта k пъти надясно и сумира получените масиви след всяко завъртане както е показано по-долу:

Примери

Вход	Изход	Коментари
3 2 4 -1 2	3 2 5 6	rotated1[] = -1 3 2 4 rotated2[] = 4 -1 3 2 sum[] = 3 2 5 6
1 2 3 1	3 1 2	rotated1[] = 3 1 2 sum[] = 3 1 2
1 2 3 4 5 3	12 10 8 6 9	rotated1[] = 5 1 2 3 4 rotated2[] = 4 5 1 2 3 rotated3[] = 3 4 5 1 2 sum[] = 12 10 8 6 9

Упътване

- След r завъртания, елементът на позиция i отива на позиция (i + r) % n.
- Масивът sum[] може да бъде изчислен с два вложени цикъла : for r = 1 … k; for i = 0 … n-1.

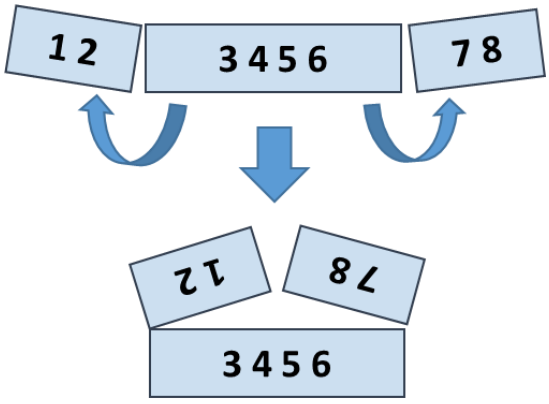
Решение

```
namespace RotateAndSum
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = 0;
            int[] nums = new int[100];
            var line = Console.ReadLine().Split(' ');
            foreach (var item in line)
            {
                nums[n++] = int.Parse(item);
            }
```

```
            int[] rotated = new int[n];
            int[] sum = new int[n];

            int rotate = int.Parse(Console.ReadLine());
            do
            {
                rotated[0] = nums[n - 1];
                for (int i = 0; i < n - 1; i++) rotated[i + 1] = nums[i];

                for (int i = 0; i < n; i++)
                {
                    nums[i] = rotated[i];
                }

                for (int i = 0; i < n; i++)
                {
                    sum[i] += rotated[i];
                }

                rotate--;
            }
            while (rotate >= 1);

            for (int i = 0; i < n; i++)
            {
                Console.Write("{0} ", sum[i]);
            }
        }
    }
}
```

Задача 3.8. Сгъни и събери

Въведете масив от 4*k цели числа, сгънете го както е указано по-долу и изведете сумата на горния и долния ред (всеки, съдържащ 2 * k цели числа):



Примери

Вход	Изход	Коментари

5 2 3 6	7 9	5 6 + 2 3 = 7 9
1 2 3 4 5 6 7 8	5 5 13 13	2 1 8 7 + 3 4 5 6 = 5 5 13 13
4 3 -1 2 5 0 1 9 8 6 7 -2	1 8 4 -1 16 14	-1 3 4 -2 7 6 + 2 5 0 1 9 8 = 1 8 4 -1 16 14

Упътване

- Създайте първия ред след сгъването: първите k числа обърнати, последвани от последните k числа, също обърнати.
- Създайте втория ред след сгъването, като вземете средните 2*k числа
- Сумирайте първи и втори ред

Решение

```
namespace FoldAndSum
{
    internal class Program
    {
        static int[] Fold(int[] nums)
        {
            var part1 = nums.Take(nums.Length / 4).ToArray();
            var part2 = nums.Skip(nums.Length / 4).Take(nums.Length / 2).ToArray();
            var part3 = nums.Skip(nums.Length * 3 / 4).Take(nums.Length /
4).ToArray();

            var a = (part1.Reverse()).Concat(part3.Reverse()).ToArray();
            var b = part2;

            return Sum(a, b);
        }

        static int[] Sum(int[] a, int[] b)
        {
            var sum = new int[a.Length];
            for (int i = 0; i < a.Length; i++)
                sum[i] = a[i] + b[i];
            return sum;
        }

        static void Main(string[] args)
        {
            int[] nums = Console.ReadLine()
                .Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries)
                .Select(int.Parse).ToArray();

            nums = Fold(nums);
            Console.WriteLine(string.Join(" ", nums));
```

```
        }
    }
}
```

Задача 3.9. Обработка на масив

Вие ще получите масив от низове и трябва да изпълните командите под тях. Вие можете да получите три команди:

- Reverse – обръща реда в масива
- Distinct – изтрива всички неуникални (повтарящи се) елементи на масива
- Replace {index} {string} – замества елемента на дадената позиция index с низ string, който ви е даден

Вход

- На първи ред, получавате масив от символни низове string array
- На втори ред ще получите броя на редове, които следват
- На следващите n реда ще получите команди

Изход

Изведете масива в следния формат:

{1st element}, {2nd element}, {3rd element} … {nth element}

Ограничения

- За разделител да се използва един инервал
- N ще е цяло число в интервала [1…100]

Примери

Вход	Изход
one one one two three four five 3 Distinct Reverse Replace 2 Hello	five, four, Hello, two, one

Решение

```
namespace ArrayProcessing
{
    class Program
    {
        static void Main(string[] args)
        {
            var array = Console.ReadLine().Split().ToArray();
            int n = int.Parse(Console.ReadLine());
            while (n > 0)
            {
                var cmd = Console.ReadLine().Split().ToArray();
                switch (cmd[0])
```

```
                {
                    case "Reverse": array = array.Reverse().ToArray(); break;
                    case "Distinct": array = array.Distinct().ToArray(); break;
                    case "Replace": array[int.Parse(cmd[1])] = cmd[2]; break;
                }
                n--;
            }
            Console.WriteLine(string.Join(", ", array));
        }
    }
}
```

Задача 3.10. Безопасна обработка на масив

Сега ние трябва да направим нашата програма, по-безопасна и по-лесна. Направете програмата да отпечата "Невалиден вход!", ако ние се опитваме да замените елемент с несъществуващ индекс или се изписва „невалидна команда", ако командата не е валидна. Също така да работи до командата "Край".

Вход

- На първи ред ще получите входния масив от низове
- На следващите редове ще получавате команди, докато не получите команда "END" за край

Изход

На края изведете масива в следния формат:

{1st element}, {2nd element}, {3rd element} … {nth element}

Ограничения

- Само един интервал да се ползва за разделител.
- n ще е цяло число integer в интервала [1…100]

Примери

Вход	Изход
one one one two three four five Distinct Reverse Replace 7 Hello Replace -5 Hello Replace 0 Hello END	Invalid input! Invalid input! Hello, four, three, two, one
Вход	**Изход**
Alpha Bravo Charlie Delta Echo Foxtrot Distinct Reverse	Invalid input! Invalid input! Alpha, Charlie, Delta, Echo

Replace 0 Charlie Reverse Replace 1 Charlie Distinct Replace 4 Charlie END	

Решение

```
namespace SaveArrayProcessing
{
    class Program
    {
        static void Main(string[] args)
        {
            var array = Console.ReadLine().Split().ToArray();
            while (true)
            {
                var cmd = Console.ReadLine().Split().ToArray();
                switch (cmd[0])
                {
                    case "END": Console.WriteLine(string.Join(", ", array)); return;
                    case "Reverse": array = array.Reverse().ToArray(); break;
                    case "Distinct": array = array.Distinct().ToArray(); break;
                    case "Replace":
                        {
                            int index = int.Parse(cmd[1]);
                            if (index < 0 || index >= array.Length)
Console.WriteLine("Invalid input!");
                            else array[index] = cmd[2];
                            break;
                        }
                    default: Console.WriteLine("Invalid input!"); break;
                }
            }
        }
    }
}
```

Задача 3.11. Множество от сумите на последните k числа

Въведете две числа n и k. Създайте и изведете следното множество от n елемента:

- Първият елемент е : 1
- Всички други елементи са = сбор от предишните k елемента (ако няма k елемента преди текущия, то да се изведе сбора на всички до момента)
- Пример: n = 9, k = 5 → 120 = 4 + 8 + 16 + 31 + 61

Примери

Вход	Изход

6 3	1 1 2 4 7 13
8 2	1 1 2 3 5 8 13 21
9 5	1 1 2 4 8 16 31 61 120

Упътване

- Използвайте масив от цели числа, в който да пазите числата.
- Инициализирайте първия елемент seq[0] = 1
- Използвайте два вложени цикъла:
  - Преминете през всички елементи в цикъл за i = 1 … n
  - Съберете елементите i-k … i-1: seq[i] = sum(seq[i-k … i-1])

Решение

```
namespace MultipleSums
{
    class Program
    {
        static int Sum(int[] nums)
        {
            int sum = 0;
            foreach (var item in nums) sum += item;
            return sum;
        }

        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            int k = int.Parse(Console.ReadLine());

            int[] seq = new int[n];
            seq[0] = 1;

            for (int i = 1; i < n; i++)
            {
                int skip = 0, take = 0;
                if (k > i)
                {
                    skip = 0;
                    take = i;
                }
                else
                {
                    skip = i - k;
                    take = k;
                }
                var next = Sum(seq.Skip(skip).Take(take).ToArray());
                seq[i] = next;
            }
```

```
            Console.WriteLine(string.Join(" ", seq));
        }
    }
}
```

Задача 3.12. Извличане на средните 1, 2 или 3 елемента

Напишете метод за извличане на средните 1,2 или 3 елемента от масив от цели числа и изведете резултата на конзолата

- n = 1 -> 1 елемент
- за четно n -> 2 елемента
- за нечетно n -> 3 елемента

Създайте програма, която чете масив от цели числа (разделени с интервал) и отпечатва средните елементи във формата показан в примерите

Примери

Вход	Изход
5	{ 5 }
2 3 8 1 7 4	{ 8, 1 }
1 2 3 4 5 6 7	{ 3, 4, 5 }
10 20 30 40 50 60 70 80	{ 40, 50 }

Упътване

- Напишете различна логика за всеки от 3 те случая (n = 1, четно n, нечетно n)
- n = 1 → взима първия елемент
- нечетно n → взема елементите n/2-1, n/2, n/2+1
- четно n → взема елементте n/2-1 и n/2

Решение

```
namespace Middle123
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] nums = Console.ReadLine().Split().Select(int.Parse).ToArray();
            int len = nums.Length;

            if (len == 1) nums = nums.Take(1).ToArray();
            else if (len % 2 == 0) nums = nums.Skip(len / 2 - 1).Take(2).ToArray();
            else nums = nums.Skip(len / 2 - 1).Take(3).ToArray();

            Console.WriteLine(string.Join(" ", nums));
        }
    }
```

```
}
```

Задача 3.13. Склад

Ще ви бъдат дадени три масиви на различни редове. Първият ще съдържа низове, които ще представляват имената на продуктите. Вторият ще съдържа големи цели числа longs и ще представляват количествата на продуктите. Третият ще съдържа дробни числа, които са цените на продуктите. След което ще бъдат дадени имена на продукти на нови редове, докато получите командата " Done". За всяко дадено име на продукт изведете :

{име на продукта} разходи: {цена}; Налично количество: {количеството}

Имената, цените и количествата на продуктите са с едни и същи индекси в 3 масива.

Вход

На трети ред вие ще получите масив с десетични числа, които представляват цените на продуктите.

Ограничения

Трите масива винаги ще имат една и съща дължина. Вие винаги ще получавате съществуващите продукти.

- На първия ред, вие ще получите масив от символни изове, които представляват имената на продуктите.
- На втори ред вие ще получите масив с дълги цели числа, които представляват количествата на продуктите.
- Третият ще съдържа дробни числа, които са цените на продуктите.

Примери

Вход	Изход
Bread Juice Fruits Lemons 10 50 20 30 2.34 1.23 3.42 1.50 Bread Juice done	Bread costs: 2.34; Available quantity: 10 Juice costs: 1.23; Available quantity: 50
Oranges Apples Nuts 1500 5000000 2000000000 2.3412 1.23 3.4250 Nuts done	Nuts costs: 3.4250; Available quantity: 2000000000

Упътване

- В C#, вие можете да намерите индекса на елемент с Array.IndexOf(array, element)
- В Java, най-лесният начин да намерите индекса на елемент (без използване на външни библиотеки) ще е да проверите целия масив

Решение

```
namespace Warehouse
{
    class Program
    {
        static int Search(string[] products, string product)
        {
            for (int index = 0; index < products.Length; index++)
            {
                if (products[index] == product)
                {
                    return index;
                }
            }
            return -1;
        }

        static void Main(string[] args)
        {
            var products = Console.ReadLine().Split().ToArray();
            var quantities =
Console.ReadLine().Split().Select(long.Parse).ToArray();
            var prices = Console.ReadLine().Split().Select(float.Parse).ToArray();

            while (true)
            {
                var cmd = Console.ReadLine();
                if (cmd == "done") break;

                var search = Search(products, cmd);
                if (search != -1)
                {
                    Console.WriteLine("{0} costs: {1}; Available quantity: {2}",
                             products[search], prices[search], quantities[search]);
                }
            }
        }
    }
}
```

Задача 3.14. Склад обновена версия

За тази задача можете да използвате вашето решение от задачата за склада. Отново ще получите 3 масиви – един с низове, с цели числа longs и с дробни числа с десетични знаци. Отново цената и количеството съответства на името, което се намира на същия индекс като име. Този път само масива, съдържащ имената и масива, съдържащ цените ще имат

същата дължина. Ако в масива с количествата няма индекс, който отговаря на името, трябва да се приеме количество 0. Освен това продуктите, които получавате след масиви ще съдържа не само низ за името, но и дълга, който е, количеството, което трябва да се поръча. Ако имате достатъчно количество, да се изчисли общата цена чрез умножаване на поръчаните количества по цената и да я отпечатате в следния формат:

{име на продукта} x {поръчано количество} струва {обща цена на поръчката}

Форматирайте цената до втория знак след десетичната запеая. Не забравяйте да намалите количеството на продукта. Ако нямате необходимото количество изведете:

We do not have enough {product name}

Вход

- На първи ред ще получите масив от низове, който съдържа имената на продуктите
- На втори ред ще получите масив от цели числа longs, който съдържа количествата на продуктите
- На трети ред ще получите масив от дробни числа , който съдържа цените на продуктите

Ограничения

- Масивите с имената и цените да са с еднаква дължина.
- Винаги се въвеждат съществуващи продукти

Примери

Вход	Изход
Bread Juice Fruits Lemons Beer 10 50 20 30 2.34 1.23 3.42 1.50 3.00 Bread 10 Juice 5 Beer 20 done	Bread x 10 costs 23.40 Juice x 5 costs 6.15 We do not have enough Beer
Tomatoes Onions Lemons 10000 2000 5.40 3.20 2.20 Tomatoes 5000 Tomatoes 5000	Tomatoes x 5000 costs 27000.00 Tomatoes x 5000 costs 27000.00 We do not have enough Tomatoes

Tomatoes 1 done	

Решение

```
namespace WarehouseUpdate
{
    internal class Program
    {
        static void Main(string[] args)
        {
            var products = Console.ReadLine().Split().ToList();
            var quantities = Console.ReadLine().Split().Select(long.Parse).ToList();
            var prices = Console.ReadLine().Split().Select(float.Parse).ToList();

            var line = Console.ReadLine();
            while (line != "done")
            {
                var command = line.Split();

                try
                {
                    var index = products.FindIndex(x => x == command[0]);
                    if (float.Parse(command[1]) > quantities[index])
                    {
                        Console.WriteLine($"We do not have enough {command[0]}");
                    }
                    else
                    {
                        Console.WriteLine($"{command[0]} x {command[1]} costs
{prices[index] * long.Parse(command[1]):f2}");
                    }
                    quantities[index] -= long.Parse(command[1]);
                }
                catch
                {
                    Console.WriteLine($"We do not have enough {command[0]}");
                }

                line = Console.ReadLine();
            }
        }
    }
}
```

Задача 3.15. Сравняване на символни масиви

Сравняваме два масива лексикографски (буква по буква). Извеждаме всеки по азбучен ред, всеки на нов ред

Примери

Вход	Изход
a b c d e f	abc def
p e t e r a n n i e	annie peter
a n n i e a n	an annie
a b a b	ab ab

Упътване

- Сравняваме първите символи на arr1[] и arr2[], ако са еднакви, сравняваме следващите и т.н.
- Ако всички символи са еднакви, по-малкия масив е по-късия
- Ако всички символи са еднакви и дължините им са равни, масивите са еднакви

Решение

```
namespace CompareStrings
{
    class Program
    {
        static void Main(string[] args)
        {
            char[] arr1 = Console.ReadLine().Split().Select(char.Parse).ToArray();
            char[] arr2 = Console.ReadLine().Split().Select(char.Parse).ToArray();
            int min = Math.Min(arr1.Length, arr2.Length);
            for (int i = 0; i < min - 1; i++)
            {
                if (arr1[i] == arr2[i]) continue;
                if (arr1[i] < arr2[i])
                {
                    Console.WriteLine(string.Join("", arr1));
                    Console.WriteLine(string.Join("", arr2));
                    return;
                }
                if (arr1[i] > arr2[i])
                {
                    Console.WriteLine(string.Join("", arr2));
                    Console.WriteLine(string.Join("", arr1));
                    return;
                }
            }
            Console.WriteLine(string.Join("", arr1));
            Console.WriteLine(string.Join("", arr2));
        }
    }
```

```
}
```

Задача 3.16. Вмъкване на елемент в сортиран масив

Даден е сортиран масив от цели числа и друго цяло число. Напишете алгоритъм, който вмъква числото в масива, така, че масива отново да е подреден.

Примери

Вход	Изход
1 2 3 7 9 4	1 2 3 4 7 9
1 2 3 4 5 0	0 1 2 3 4 5

Упътване

- Сравняваме числото със средния елемент, ако е по-малко търсим в първата половина на масива (надолу), иначе – във втората (нагоре)
- Повтаряме горното правило докато масива, в който търсим има неповече от два елемента
- Мястото на числото е или преди по-малкия, или между двата или след по-големия елемент.
- Вмъкваме числото

Решение

```
namespace InsertIntoArray
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] A = Console.ReadLine().Split().Select(int.Parse).ToArray();
            int n = int.Parse(Console.ReadLine());
            int[] B = new int[A.Length + 1];
            int placed = 0;
            for (int i = 0; i < B.Length; i++)
            {
                if (n >= A[i - placed] || placed == 1)
                {
                    B[i] = A[i - placed];
                }
                else
                {
                    B[i] = n;
                    placed = 1;
                }
            }
            Console.WriteLine(string.Join(" ", B));
        }
```

```
    }
}
```

Задача 3.17. Търсене на елемент в сортиран масив

Даден е сортиран масив от цели числа и друго цяло число. Напишете алгоритъм, който извежда "Yes" ако елемента се намира в масива и "No" ако елемента не се намира в масива.

Примери

Вход	Изход
1 2 3 7 9 7	Yes
1 2 3 4 5 0	No

Упътване

Алгоритъмът е подобен на предишния

Решение

```
namespace ArraySearch
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] nums = Console.ReadLine().Split().Select(int.Parse).ToArray();
            int n = int.Parse(Console.ReadLine());
            bool found = false;
            for (int i = 0; i < nums.Length; i++)
            {
                if (nums[i] == n)
                {
                    found = true;
                    break;
                }
            }
            Console.WriteLine(found ? "Yes" : "No");
        }
    }
}
```

Задача 3.18. Сливане на подредени масиви

Създайте програма, която по зададени два подредени във възходящ ред масиви от цели числа, създава трети, който отново е подреден

Примери

Вход	Изход
1 2 3 7 9 2 4 5 7 8	1 2 2 3 4 5 7 7 8 9

1 2 3 4 5 1 7 9 10	1 1 2 3 4 5 7 9 10

Упътване

- Използвайте факта, че масивите са подредени
- Ако един елемент от единия масив е по-малък от елемент другия масив, то вземаме за последващата проверка елемент от същия масив иначе прилагаме правилото за другия масив

Решение

```
namespace ArrayMerge
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] A = Console.ReadLine().Split().Select(int.Parse).ToArray();
            int[] B = Console.ReadLine().Split().Select(int.Parse).ToArray();
            int[] C = new int[A.Length + B.Length];

            for (int i = 0; i < A.Length; i++)
            {
                C[i] = A[i];
            }

            for (int i = A.Length; i < A.Length + B.Length; i++)
            {
                C[i] = B[i - A.Length];
            }

            for (int i = 0; i < C.Length; i++)
            {
                for (int j = 0; j < C.Length; j++)
                {
                    if (C[i] < C[j])
                    {
                        int temp = C[i];
                        C[i] = C[j];
                        C[j] = temp;
                    }
                }
            }

            Console.WriteLine(string.Join(" ", C));
        }
    }
}
```

Задача 3.19. Сортиране

От клавиатурата се въвежда масив от цели числа. Сортирайте го в низходящ ред. Изведете резултата на един ред.

Примери

Вход	Изход
1 5 -11 35 -3	-11 -3 1 5
84 2 90 110 34 6	2 6 34 84 110

Упътване

- Намерете най-малкия елемент от масива и разменете мястото му с първия
- Повтаряйте горната стъпка с елементите от втория до последния и т.н

Решение

```
namespace Sorting
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] A = Console.ReadLine().Split().Select(int.Parse).ToArray();

            for (int i = 0; i < A.Length; i++)
            {
                for (int j = 0; j < A.Length; j++)
                {
                    if (A[i] < A[j])
                    {
                        int temp = A[i];
                        A[i] = A[j];
                        A[j] = temp;
                    }
                }
            }

            Console.WriteLine(string.Join(" ", A));
        }
    }
}
```

Задача 3.20. Въвеждане на списък от конзолата чрез 1 ред

Въведете списък от цели числа и го изведете в конзолата. Елементите на списъка ще получите от единствен ред, разделени с интервали.

1. Добавете нов конзолен проект и задайте име **ListInputOutputLine**.
2. Въведете списъка от числа от един ред, чрез следния код:

```
List<int> nums = Console.ReadLine().Split().Select(int.Parse).ToList();
```

3. Изведете списъка по начин подобен на предната задача

Примери

Вход	Изход

1 2 3 4 5 6	nums[0] = 1 nums[1] = 2 nums[2] = 3 nums[3] = 4 nums[4] = 5 nums[5] = 6

Решение

```
namespace OneCommandLineToList
{
    class Program
    {
        static void Main(string[] args)
        {
            List<int> list = Console.ReadLine().Split().Select(int.Parse).ToList();

            Console.WriteLine(string.Join(", ", list));
        }
    }
}
```

Задача 3.21. Списък от имена

Въведете списък от имена на хора и го изведете в конзолата в обратен ред. Елементите на списъка ще получите от единствен ред, разделени с интервали. Изведете имената на единствен ред, така че след всяко да стои знак ;

Примери

Вход	Изход
Ivan Maria Dimitar Simona Petya	Ivan; Maria; Dimitar; Simona; Petya

Решение

```
namespace NamesList
{
    class Program
    {
        static void Main(string[] args)
        {
            var list = Console.ReadLine().Split(',').ToList();

            foreach (var item in list)
            {
                var parts = item.Trim().Split(' ').ToList();
                Console.WriteLine("{0} {1}", parts[1], parts[0]);
            }
        }
    }
}
```

Задача 3.22. Списък от имена 2

Въведете списък от имена на хора и го изведете в конзолата в обратен ред. Елементите на списъка ще получите от единствен ред, разделени със

запетаи. Всеки елемент ще представлява име и фамилия. Изведете имената на всеки човек на отделен ред, като първо трябва да изведете фамилията, след което личното име.

Примери

Вход	Изход
Ivan Dimitrov, Maria Ivanova, Dimitar Petrov	Dimitrov Ivan Ivanova Maria Petrov Dimitar

Решение

```
namespace NamesList2
{
    class Program
    {
        static void Main(string[] args)
        {
            var list = Console.ReadLine().Split(',').Reverse().ToList();

            foreach (var item in list)
            {
                var parts = item.Trim().Split(' ').ToList();
                Console.WriteLine("{0} {1}", parts[1], parts[0]);
            }
        }
    }
}
```

Задача 3.23. Списък от четни числа

Въведете списък от цели числа и изведете четните числа от списъка на един ред в конзолата. Елементите на списъка ще получите от единствен ред, разделени с интервали.

Примери

Вход	Изход
3 4 8 5 7 5 2 1	4 8 2
1 2 4 3	2 4
7 2 8 3 5 9 7 3	2 8

Решение

```
namespace EvenNumbersList
{
    class Program
    {
        static void Main(string[] args)
        {
            List<int> nums = Console.ReadLine().Split().Select(int.Parse).ToList();
            for (int index = 0; index < nums.Count; index++)
            {
```

```
                if (nums[index] % 2 == 0)
                    Console.Write("{0} ", nums[index]);
            }
        }
    }
}
```

Задача 3.24. Списък от крайности

Въведете списък от цели числа и изведете тези от тях, които са равни на минималния или максималния елемент.

Примери

Вход	Изход
5 4 8 5 7 8 2 1	1 8 8
1 1 1	1 1 1
4 2 8 3 5 9 2 3	2 2 9

Решение

```
namespace ExtremesList
{
    class Program
    {
        static void Main(string[] args)
        {
            var nums = Console.ReadLine().Split().Select(int.Parse).ToList();
            List<int> result = new List<int>();

            int min = nums[0], max = nums[0];
            foreach (var item in nums)
            {
                if (item < min) min = item;
                if (item > max) max = item;
            }

            for (int index = 0; index < nums.Count; index++)
            {
                if (nums[index] == min || nums[index] == max)
                {
                    result.Add(nums[index]);
                }
            }

            result.Sort();
            Console.WriteLine(string.Join(" ", result));
        }
    }
}
```

Задача 3.25. Максимална поредица еднакви числа

Въведете списък от цели числа и намерете най-дългата поредица от еднакви елементи. Ако съществуват няколко, отпечатайте най-лявата.

Примери

Вход	Изход
3 4 4 5 5 5 2 2	5 5 5
7 7 4 4 5 5 3 3	7 7
1 2 3 3	3 3

Подсказки

- Обходете позициите p отляво надясно и пазете началото и дължината на текущата поредица от еднакви числа приключаваща с p.
- Също така пазете текущата най-добра (най-дълга) поредица (bestStart – позицията, на която започва, както и bestLength – нейната дължина) и я обновявайте след всяка стъпка

Решение

```
namespace LongestSequenceOfSameNumbers
{
    class Program
    {
        static void Main(string[] args)
        {
            List<int> numbers = Console.ReadLine().Split('
').Select(int.Parse).ToList();

            int seq = 1, maxSeq = 1, mostCommonNumber = 0;

            for (int i = 0; i < numbers.Count - 1; i++)
            {
                if (numbers[i] == numbers[i + 1]) seq++;
                else seq = 1;
                if (seq > maxSeq)
                {
                    maxSeq = seq;
                    mostCommonNumber = numbers[i];
                }
            }

            for (int i = 0; i < maxSeq; i++)
            {
                Console.Write(mostCommonNumber + " ");
            }
        }
    }
}
```

Задача 3.26. Сума на обърнати числа

Напишете програма, която прочита поредица от цели числа, преобръща техните цифри и ги сумира.

Примери

Вход	Изход	Пояснения
123 234 12	774	321 + 432 + 21 = 774
12 12 34 84 66 12	220	21 + 21 + 43+ 48 + 66 + 21 = 220
120 1200 12000	63	21 **21 + 21 = 63**

Решение

```
namespace SumOfReversedNumbers
{
    class Program
    {
        public static string ReverseString(string s)
        {
            char[] charArray = s.ToCharArray();
            Array.Reverse(charArray);
            return new string(charArray);
        }

        static void Main(string[] args)
        {
            var nums = Console.ReadLine().Split().ToList();
            var sum = 0;
            foreach (var item in nums)
                sum += int.Parse(ReverseString(item));
            Console.WriteLine(sum);
        }
    }
}
```

Задача 3.27. Премахни числото

Въведете списък от цели числа и премахнете всички срещания в списъка на последното число. Елементите на списъка ще получите от единствен ред, разделени с интервали.

Примери

Вход	Изход
3 4 1 5 1 5 2 1	3 4 5 5 2
7 3 8 3 5 3 7 3	7 8 5 7
2 2 8 2 5 2 3 2	8 5 3

Подсказки

- Извлечете стойността на последния елемент. Той се намира на индекс равен на броя на елементите минус 1. Броят на елементите може да разберете чрез Count
- Докато елементът съществува, премахвайте първото му срещане чрез Remove

Решение

```
namespace RemoveTheNumber
{
    class Program
    {
        static void Main(string[] args)
        {
            List<int> nums = Console.ReadLine().Split().Select(int.Parse).ToList();
            int number = nums[nums.Count - 1];
            while (nums.Contains(number))
            {
                nums.Remove(number);
            }
            Console.WriteLine(string.Join(" ", nums));
        }
    }
}
```

Задача 3.28. Изтриване на отрицателни елементи

Въведете списък от цели числа, премахнете всички отрицателни числа от него и го изведете на конзолата в обратен ред. В случай, че в списъка не са останали елементи, изведете "empty".

Примери

Вход	Изход
10 -5 7 9 -33 50	50 9 7 10
7 -2 -10 1	1 7
-1 -2 -3	Empty

Подсказки

- Създайте нов празен списък за получения като резултат списък
- Обходете въведения списък отзад напред. Проверете всеки елемент и добавете неотрицателните елементи към списъка за резултат
- Накрая, изведете списъка резултат на единствен ред, разделен с интервали.

Решение

```
namespace RemoveNegativeElements
{
    class Program
    {
        static void Main(string[] args)
```

```
        {
            List<int> nums = Console.ReadLine().Split().Select(int.Parse).ToList();

            for (int index = 0; index < nums.Count; index++)
            {
                if (nums[index] < 0)
                {
                    nums.RemoveAt(index);
                    index--;
                }
            }

            Console.WriteLine(string.Join(" ", nums));
        }
    }
}
```

Задача 3.29. Сливане на списъци

Напишете програма, която слива няколко списъка от числа.

- Списъците се разделят от '|'.
- Стойностите се разделят от интервали (' ', един или няколко)
- Подредете списъците отзад напред, а техните стойности отляво надясно.

Примери

Вход	Изход
1 2 3 |4 5 6 | 7 8	7 8 4 5 6 1 2 3
7 | 4 5|1 0| 2 5 |3	3 2 5 1 0 4 5 7
1| 4 5 6 7 | 8 9	8 9 4 5 6 7 1

Подсказки

- Създайте нов празен списък за резултатите.
- Отделете входа чрез '|' така че да се получи списък от низове.
- Обходете получения списък отдясно наляво.
  - За всеки низ в списъка: отделете елементите му чрез знака за интервал
  - Всеки един елемент, който е непразен низ, трябва да бъде добавен към списъка с резултата
- Изведете списъка с резултата

Решение

```
namespace MergeLists
{
    class Program
    {
        static void Main(string[] args)
        {
            var lists = Console.ReadLine().Split('|').ToList();
            List<int> result = new List<int>();
```

```
            for (int index = lists.Count - 1; index >= 0; index--)
            {
                List<string> nums = lists[index].Split(' ').ToList();

                for (int index2 = 0; index2 < nums.Count; index2++)
                {
                    if (nums[index2] != "")
                    {
                        result.Add(int.Parse(nums[index2]));
                    }
                }
            }

            Console.WriteLine(string.Join(" ", result));
        }
    }
}
```

Задача 3.30. Бомбички

Напишете програма, която въвежда поредица от числа и специално число - бомбичка с определена сила. Вашата задача е да детонирате всяко срещане на специалното число бомба и според нейната сила нейните съседи отляво и отдясно. Детонациите се изпълняват отляво надясно и всички детонирани числа изчезват. Най-накрая изведете сумата от оставащите елементи в поредицата.

Примери

Вход	Изход	Коментари
1 2 2 4 2 2 2 9 4 2	12	Бомбичката е 4 със сила 2. След детонацията остават [1, 2, 9] със сума 12.
1 4 4 2 8 9 1 9 3	5	Бомбичката е 9 със сила 3. След детонацията оставаме с поредицата [1, 4], която има сума 5. Понеже 9 има само 1 съсед отдясно, ние го премахваме
1 7 7 1 2 3 7 1	6	Детонациите се изпълняват отляво надясно. Не можем да детонираме второто срещане на 7, понеже то вече е унищожено от първата детонация. Остават [1, 2, 3]. Тяхната сума е 6.
1 1 2 1 1 1 2 1 1 1 2 1	4	Оцветените числа изчезват в две последователни детонации. Оставащата предица е [1, 1, 1, 1]. Нейната сума е 4.

Решение

```
namespace Bombs
{
    class Program
    {
        static void Main(string[] args)
        {
            List<int> numbers = Console.ReadLine().Split('
').Select(int.Parse).ToList();
            int[] bombPower = Console.ReadLine().Split('
').Select(int.Parse).ToArray();
            int bomb = bombPower[0];
            int power = bombPower[1];

            while (numbers.Contains(bomb))
            {
                int area = numbers.IndexOf(bomb);

                if (area - power < 0 && area + power >= numbers.Count)
                {
                    numbers.Clear();
                    break;
                }

                for (int i = area - power; i <= area + power; i++)
                {
                    if (i < 0) continue;
                    if (area - power >= numbers.Count) break;
                    numbers.RemoveAt(area - power);
                }
            }

            int sum = 0;
            foreach (var item in numbers)
            {
                sum += item;
            }

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

Задача 3.31. Сортиране на числа

Въведете списък от цели числа и го сортирайте.

Примери

Вход	Изход
8 2 7 3	2 <= 3 <= 7 <= 8
1 1	1 <= 1
2 4 -9	-9 <= 2 <= 4

1 -0.5	-0.5 <= 1

Решение

```
namespace NumbersSort
{
    class Program
    {
        static void Main(string[] args)
        {
            List<double> nums =
Console.ReadLine().Split().Select(double.Parse).ToList();
            nums.Sort();
            Console.WriteLine(string.Join(" <= ", nums));
        }
    }
}
```

Задача 3.32. Числа квадрати

Въведете списък от цели числа и изведете всички числа квадрати от списъка в намаляващ ред. Число квадрат е цяло число, което е квадрат на друго цяло число. Например, 1, 4, 9, 16 са числа квадрати.

Примери

Вход	Изход
3 16 4 5 6 8 9	16 9 4
12 1 9 4 16 8 25 49 16	49 25 16 16 9 4 1

Подсказки

- За да разберете дали едно цяло число е "число квадрат", проверете дали неговия корен квадратен е цяло число (такова че да няма дробна част):
- За да подредите списъка от резултати в намаляващ ред използвайте сортиране с ламбда функция:

Решение

```
namespace SquareNumbers
{
    internal class Program
    {
        static void Main(string[] args)
        {
            var numbers = Console.ReadLine().Split().Select(int.Parse).ToList();
            var squareNums = numbers.Where(x => Math.Sqrt(x) % 1 == 0).ToList();
            squareNums.Sort((a, b) => b.CompareTo(a));
            Console.WriteLine(string.Join(" ", squareNums));
        }
    }
}
```

Задача 3.33. Брой на числа

Въведете списък от цели числа в интервала [0...1000] и ги изведете в нарастващ ред заедно с броя на срещанията им.

Примери

Вход	Изход
8 2 2 8 2 2 3 7	2 -> 4 3 -> 1 7 -> 1 8 -> 2
10 8 8 10 10	8 -> 2 10 -> 3

Подсказки

1. Въведете елементите в масива от цели числа **nums[]**. Например: {8, 2, 2, 8, 2, 2, 3, 7}.
2. Сортирайте **nums[]** в нарастващ ред: {2, 2, 2, 2, 3, 7, 8, 8}. Сега намерете всички подредици от едни и същи числа.
3. Обходете числата отляво надясно. Пребройте колко пъти се среща всяко число
   - Започнете с **count** = **1**.
   - Докато следващото число отдясно е същото като сегашното, увеличавайте **count** и продължете към следващото число.
   - Когато числото отдясно е различно (или няма друго число), изведете текущия елемент и неговия брой.
   - Продължете да обхождате от следващото число отдясно.

Решение

```
namespace NumbersCount
{
    class Program
    {
        static void Main(string[] args)
        {
            var nums = Console.ReadLine().Split().Select(int.Parse).ToList();

            var counts = new int[nums.Max() + 1];
            foreach (var num in nums)
            {
                counts[num]++;
            }

            for (int i = 0; i < counts.Length; i++)
            {
                if (counts[i] > 0)
                {
                    Console.WriteLine($"{i} -> {counts[i]}");
                }
```

```
            }
        }
    }
}
```

Задача 3.34. Сума на съседни еднакви числа

Напишете програма, която сумира всички съседни еднакви числа в списък от цели числа, започвайки отляво надясно.

- **След като две числа са сумирани, полученият резултат може да бъде равен на някой от другите му съседи, което означава, че също трябва да се сумира (вижте примерите).**
- **Винаги сумирайте най-левите две еднакви числа (ако има няколко двойки от еднакви числа).**

Примери

Вход	Изход	Обяснение
3 3 6 1	12 1	3 3 6 1 → 6 6 1 → 12 1
8 2 2 4 8 16	16 8 16	8 2 2 4 8 16 → 8 4 4 8 16 → 8 8 8 16 → 16 8 16
5 4 2 1 1 4	5 8 4	5 4 2 1 1 4 → 5 4 2 2 4 → 5 4 4 4 → 5 8 4

Подсказки

1. Въведете числата и създайте списък от числа.
2. Намерете двете най-леви съседни еднакви клетки.
3. Заменете ги с тяхната сума.
4. Повторете (1) и (2) докато не остават две съседни еднакви клетки.
5. Изведете обработения списък.

Решение

```
namespace EqualNumbersSum
{
    internal class Program
    {
        static bool HasEquals(List<int> numbers)
        {
            for (int i = 0; i < numbers.Count - 1; i++)
            {
                if (numbers[i] == numbers[i + 1])
                {
                    return true;
                }
            }
            return false;
        }

        static void Main(string[] args)
        {
            List<int> numbers =
Console.ReadLine().Split().Select(int.Parse).ToList();

            while (HasEquals(numbers))
```

```
            {
                for (int i = 0; i < numbers.Count - 1; i++)
                {
                    if (numbers[i] == numbers[i + 1])
                    {
                        numbers.Insert(i, numbers.Skip(i).Take(2).Sum());
                        numbers.RemoveRange(i + 1, 2);
                        break;
                    }
                }
            }
            Console.WriteLine(string.Join(" ", numbers));
        }
    }
}
```

Задача 3.35. Отделяне по регистър на дума

Въведете text, след което го разделете към думи и ги разпредете в 3 списъка.

- Думи с малки букви като "programming", "at" и"databases" – съдържащи се само от малки букви.
- Думи с големи букви като "PHP", "JS" and "SQL" – съдържат само големи бувки.
- Смесени думи като "C#", "CodeCamp" и "Java" – всички други.

Използвайте следните разделители между думите: , ; : . ! ( ) " ' \ / [ ] интервал

Изведете трите списъка, както е показано в примера.

Примери

Вход	Изход
Learn programming at CodeCamp: Java, PHP, JS, HTML 5, CSS, Web, C#, SQL, databases, AJAX, etc.	Lower-case: programming, at, databases, etc Mixed-case: Learn, CodeCamp, Java, 5, Web, C# Upper-case: PHP, JS, HTML, CSS, SQL, AJAX

Подсказки

- Отделете входния текст чрез използваните по-горе разделители.
- Обработете получения списък от думи една по една.
- Създайте 3 списъка от думи (в началото празни): думи с малки букви, думи с големи букви, думи със смесени букви.
- Проверете всяка дума и я разпределете към някой от трите списъка:
  - Пребройте всички малки букви и големи букви.
  - Ако всичките букви са малки, добавете думата към списък на думите с малки букви
  - Ако всичките букви са големи, добавете думата към списък на думите с големи букви

- В противен случай се смята, че думата е със смесени букви → добавяме я към списъка на думите със смесени букви.

- Изведете получените списъци, както е показано в списъка горе.

Решение

```
namespace WordsRegister
{
    internal class Program
    {
        static bool Upper(string word)
        {
            foreach (var character in word)
            {
                if (Char.IsLower(character))
                {
                    return false;
                }
            }
            return true;
        }

        static bool Lower(string word)
        {
            foreach (var character in word)
            {
                if (Char.IsUpper(character))
                {
                    return false;
                }
            }
            return true;
        }

        static void Main(string[] args)
        {
            var words = Console.ReadLine().Split();

            List<string> LowerCase = new List<string>();
            List<string> MixedCase = new List<string>();
            List<string> UpperCase = new List<string>();

            foreach (var word in words)
            {
                if (Lower(word)) LowerCase.Add(word);
                else if (Upper(word)) UpperCase.Add(word);
                else MixedCase.Add(word);
            }

            Console.WriteLine("Lower-case: {0}", string.Join(", ", LowerCase));
            Console.WriteLine("Mixed-case: {0}", string.Join(", ", MixedCase));
            Console.WriteLine("Upper-case: {0}", string.Join(", ", UpperCase));
        }
    }
}
```

Задача 3.36. Променлив списък

Напишете програма, която въвежда списък от цели числа от конзолата и получава команди, които манипулират списъка. Вашата програма може да получава следните команди:

- Delete {елемент} – изтрива всички елементи в списъка, които са равни на дадения елемент
- Insert {елемент} {позиция} – вмъква елемент на дадената позиция

Програмата трябва да приключва, когато получи команда Odd или Even. Ако програмата получи Odd, то извежда всички нечетни числа в списъка отделени с единствен интервал, иначе извеждаме по същия начин всички четни числа.

Примери

Вход	Изход
1 2 3 4 5 5 5 6 Delete 5 Insert 10 1 Delete 5 Odd	1 3

Вход	Изход
20 12 4 319 21 31234 2 41 23 4 Insert 50 2 Insert 50 5 Delete 4 Even	20 12 50 50 31234 2

Решение

```
namespace VariableList
{
    public class Program
    {
        static void Main(string[] args)
        {
            List<int> numbers =
Console.ReadLine().Split().Select(int.Parse).ToList();
            string[] command;
            while(true)
            {
                command = Console.ReadLine().Split().ToArray();
                switch (command[0])
                {
                    case "Insert":
                        numbers.Insert(int.Parse(command[2]),
int.Parse(command[1]));
                        break;

                    case "Delete":
                        numbers.RemoveAll(number => number ==
int.Parse(command[1]));
                        break;

                    case "Odd":
                        Console.WriteLine(string.Join(" ", numbers.Where(x => x % 2
!= 0)));
```

```
                    return;

                case "Even":
                    Console.WriteLine(string.Join(" ", numbers.Where(x => x % 2
== 0)));
                    return;

            }
        }
    }
}
```

Задача 3.37. Търсене на число

На първия ред се въвежда списък от цели числа. На следващия ред, ще получите списък с точно три числа. Първото от тях показва броя на елементите, които трябва да вземете от списъка (считано от първия елемент). Второто число показва броя на елементите, които трябва да изтриете от елементите, които взехте (считано от първия елемент). Последното число е това, което търсим в получения списък след манипулациите. Ако това число е в списъка, извеждаме: "YES!", в противен случай "NO!"

Примери

Вход	Изход
1 2 3 4 5 6 5 2 3	YES!

Вход	Изход
12 412 123 21 654 34 65 3 23 7 4 21	NO!

Решение

```
namespace NumbersSearch
{
    public class Program
    {
        static void Main(string[] args)
        {
            List<int> numbers =
Console.ReadLine().Split().Select(int.Parse).ToList();

            int[] command = Console.ReadLine().Split().Select(int.Parse).ToArray();

            numbers = numbers.Take(command[0]).Skip(command[1]).ToList();

            if (numbers.Contains(command[2]))
            {
                Console.WriteLine("YES!");
            }
            else
            {
                Console.WriteLine("NO!");
            }
        }
    }
```

```
}
```

Задача 3.38. Най-дълга нарастваща под редица (Longest Increasing Subsequence – LIS)

Въведете списък от цели числа и намерете най-дългата растяща под редица (LIS). Ако има няколко такива, изведете най-лявата.

Примери

Вход	Изход
1	1
7 3 5 8 -1 0 6 7	3 5 6 7
1 2 5 3 5 2 4 1	1 2 3 5
0 10 20 30 30 40 1 50 2 3 4 5 6	0 1 2 3 4 5 6
11 12 13 3 14 4 15 5 6 7 8 7 16 9 8	3 4 5 6 7 8 16
3 14 5 12 15 7 8 9 11 10 1	3 5 7 8 9 11

Подсказки

- Нека имаме **n** числа в списъка **nums[0…n-1]**.
- Нека **len[p]** показва дължината на най-дългата растяща под редица (LIS) завършваща в позиция **p**.
- Във for-цикъл, трябва да изчислим **len[p]** за **p** = **0** … **n-1** както следва:
  - Нека **left** е най-лявата позицията наляво от **p** (**left** < **p**), така щото **len[left]** да е колкото се може по-голямо.
  - Тогава, **len[p]** = **1** + **len[left]**. Ако **left** не съществува, **len[p]** = **1**.
  - Също така, запазете **prev[p]** = **left** (запазваме си в **prev[]** предната позицията, която сме използвали за да получим най-добрата дължина за позицията **p**).
- Веднъж щом стойностите на **len[0…n-1]** са изчислени, върнете най-дългата растяща под редица започвайки от **p** така че **len[p]** да е максималното и се върнете назад чрез **p** = **prev[p]**.
- Тази таблица илюстрира изчисленията за последния пример:

index	0	1	2	3	4	5	6	7	8	9	10
nums[]	3	14	5	12	15	7	8	9	11	10	1
len[]	1	2	2	3	4	3	4	5	6	6	1
prev[]	-1	0	0	2	3	2	5	6	7	7	-1
LIS	{3}	{3,14}	{3,5}	{3,5,12}	{3,5, 12,15}	{3,5,7}	{3,5, 7,8}	{3,5, 7,8,9}	{3,5,7, 8,9,11}	{3,5,7, 8,9,10}	{1}

Решение

```
namespace LongestIncreasingSubsequence
{
```

```
    class Program
    {
        static void Main(string[] args)
        {
            uint[] arr = Console.ReadLine().Split(' ').Select(uint.Parse).ToArray();
            uint longestSequence = 0, bestNumber = 0;
            for (int i = 0; i < arr.Length; i++)
            {
                uint currentSequence = 1;
                for (int j = i + 1; j < arr.Length; j++)
                {
                    if (arr[j] == arr[i] + currentSequence++)
                        ;
                    else
                    {
                        currentSequence--;
                        break;
                    }
                }
                if (currentSequence > longestSequence)
                {
                    longestSequence = currentSequence;
                    bestNumber = arr[i];
                }
            }
            for (int i = 0; i < longestSequence; i++)
            {
                Console.Write($"{bestNumber++} ");
            }
        }
    }
}
```

Задача 3.39. Списъчен манипулатор

Напишете програма, която въвежда списък от цели числа от конзолата и списък от команди, които се изпълняват върху списъка. Командите са както следва:

- add <индекс> <елемент> – вмъква елемент на зададената позиция (елементите надясно от тази позиция включително се изместват надясно).
- addMany <индекс> <елемент 1> <елемент 2> ... <елемент n> – добавя множество от елементи на дадената позиция.
- contains <елемент> – изпечатва индекса на първото срещане на зададения елемент (ако съществува) в списъка или -1, ако елемента не е открит.
- remove <индекс> – премахва елемента, намиращ се на зададената позиция
- shift <позиции> – измества всеки елемент от списъка съответния брой позиции наляво (с ротация).
    - Например, [1, 2, 3, 4, 5] -> shift 2 -> [3, 4, 5, 1, 2]
- sumPairs – сумира елементите на всички двойки в списъка (първа + втора, трета + четвърта, ...).

- Например, [1, 2, 4, 5, 6, 7, 8] -> [3, 9, 13, 8].
- print – спира да получава повече команди и извежда последното състояние на списъка.

Примери

Вход	Изход
1 2 4 5 6 7 add 1 8 contains 1 contains -3 print	0 -1 [1, 8, 2, 4, 5, 6, 7]
1 2 3 4 5 addMany 5 9 8 7 6 5 contains 15 remove 3 shift 1 print	-1 [2, 3, 5, 9, 8, 7, 6, 5, 1]
2 2 4 2 4 add 1 4 sumPairs print	[6, 6, 6]
1 2 1 2 1 2 1 2 1 2 1 2 sumPairs sumPairs addMany 0 -1 -2 -3 print	[-1, -2, -3, 6, 6, 6]

Решение

```
namespace ListManipulator
{
    public static class Program
    {
        public static List<T> ShiftLeft<T>(this List<T> list, int shiftBy)
        {
            if (list.Count <= shiftBy)
            {
                return list;
            }

            var result = list.GetRange(shiftBy, list.Count - shiftBy);
            result.AddRange(list.GetRange(0, shiftBy));
            return result;
        }

        static void Main(string[] args)
        {
```

```
            List<int> numbers =
Console.ReadLine().Split().Select(int.Parse).ToList();
            string[] command;
            while(true)
            {
                command = Console.ReadLine().Split().ToArray();
                switch (command[0])
                {
                    case "add":
                        numbers.Insert(int.Parse(command[1]),
int.Parse(command[2]));
                        break;

                    case "addMany":
                        var items = command.Skip(2).Select(int.Parse).ToList();
                        numbers.InsertRange(int.Parse(command[1]), items);
                        break;

                    case "contains":
                        var index = numbers.Find(number => number ==
int.Parse(command[1]));
                        if (!numbers.Contains(int.Parse(command[1]))) index = -1;
                        Console.WriteLine(index);
                        break;

                    case "remove":
                        numbers.RemoveAt(int.Parse(command[1]));
                        break;

                    case "shift":
                        numbers = ShiftLeft(numbers, int.Parse(command[1]));
                        break;

                    case "sumPairs":
                        List<int> sums = new List<int>();
                        for (int i = 0; i < numbers.Count; i++)
                        {
                            var pair = numbers.Skip(i * 2).Take(2);
                            if (pair.Count() == 0) break;
                            sums.Add(pair.Sum());
                        }
                        numbers = sums;
                        break;

                    case "print":
                        Console.WriteLine(string.Join(" ", numbers));
                        return;
                }
            }
        }
    }
}
```

Тема 4. Методи и дебъгване

Задача 4.1. Празна касова бележка

Създайте метод, който отпечатва празна касова бележка. Методът трябва да извиква три други метода: един за отпечатване на хедъра, един за основната част на бележката и един за футъра.

Хедърът трябва да съдържа следния текст:	CASH RECEIPT ------------------------------
Основната част на бележката съдържа текста:	Charged to_____ Received by_____
Ето го и текста на футъра:	------------------------------ © BG

Пример

Изход
CASH RECEIPT ------------------------------ Charged to_____ Received by_____ ------------------------------ © BG

Подсказки

1. Първо създайте метод без параметри за отпечатването на хедъра. Дефиницията му започва със static void. Дайте му смислено име, например "PrintReceiptHeader" и напишете кода, който ще изпълнява този метод:
2. Направете същото и за същинската част и футъра на касовата бележка.
3. Създайте метод, който ще извиква тези три метода в правилния ред. И на него дайте смислено и описателно име, примерно "PrintReceipt" и напишете неговия програмен код:
4. За отпечатване на "©" използвайте Уникод-а "\u00A9"
5. Извиквайте методът PrintReceipt от main метода.

Решение

```
namespace EmptyReceipt
{
    class Program
    {
        private static void PrintReceipt()
        {
            PrintHeader();
            PrintBody();
            PrintFooter();
        }

        private static void PrintHeader()
```

```
        {
            Console.WriteLine("CASH RECEIPT");
            Console.WriteLine("------------------------------");
        }

        private static void PrintBody()
        {
            Console.WriteLine("Charged to____________________");
            Console.WriteLine("Received by___________________");
        }

        private static void PrintFooter()
        {
            Console.WriteLine("------------------------------");
            Console.WriteLine("\u00A9 SoftUni");
        }

        static void Main(string[] args)
        {
            PrintReceipt();
        }
    }
}
```

Задача 4.2. Знак на цяло число

Създайте метод, отпечатващ знака на цяло число n.

Пример

Вход	Изход
2	The number 2 is positive.
-5	The number -5 is negative.
0	The number 0 is zero.

Подсказки

1. Създайте метод с описателно име като "PrintSign". Методът трябва да получава един параметър от тип int.
2. Изградете и тялото на метода, като обработите трите случая:
    a. Ако числото е по-голямо от нула
    b. Ако числото е по-малко от нула
    c. И ако числото е равно на нула
3. Извикайте новосъздадения метод от метода main.

Решение

```
namespace IntegerSign
{
    class Program
    {
        static void PrintSign(int number)
        {
```

```
            if (number > 0)
                Console.WriteLine("The number {0} is positive", number);
            else if (number < 0)
                Console.WriteLine("The number {0} is negative.", number);
            else
                Console.WriteLine("The number {0} is zero.", number);
        }

        static void Main(string[] args)
        {
            PrintSign(int.Parse(Console.ReadLine()));
        }
    }
}
```

Задача 4.3. Отпечатване на триъгълник

Създайте метод за отпечатване на триъгълници както е показано по-долу:

Примери

Вход	Изход
3	1 1 2 1 2 3 1 2 1
4	1 1 2 1 2 3 1 2 3 4 1 2 3 1 2 1

Решение

```
namespace TrianglePrint
{
    class Program
    {
        static void PrintLine(int start, int end)
        {
            for (int i = start; i <= end; i++)
            {
                Console.Write(i + " ");
            }
            Console.WriteLine();
        }

        static void PrintTriangle(int n)
```

```
        {
            for (int line = 1; line <= n; line++)
                PrintLine(1, line);

            for (int line = n - 1; line >= 1; line--)
                PrintLine(1, line);
        }

        static void Main(string[] args)
        {
            PrintTriangle(int.Parse(Console.ReadLine()));
        }
    }
}
```

Задача 4.4. Изчертаване на запълнен квадрат

Изчертайте на конзолата запълнен квадрат с дължина на страната n като в примера:

Пример

Вход	Изход
4	-------- -\/\/\/- -\/\/\/- --------

Подсказки

1. Прочетете входните данни
2. Създайте метод, който ще печати най-горния и най-долния ред (тъй като те са еднакви). Не забравяйте да му дадете описателно име и като параметър да му подадете някаква дължина.
    a. Вместо цикъл може да използвате командата "new string", която създава нов текст, съставен от символ, повторен определен брой пъти:
3. Сега създайте метод, който ще отпечатва средните редове. Е, ясно е, ще го наречете предполагам "PrintMiddleRow" ☺
4. Използвайте методите, които току-що създадохте, за изчертаването на квадрата:

Решение

```
namespace DrawFillRectangle
{
    class Program
    {
        static void PrintHeaderRow(int n)
        {
            Console.WriteLine(new string('-', 2 * n));
        }

        static void PrintMiddleRow(int n)
```

```
        {
            Console.Write('-');
            for (int i = 1; i < n; i++)
            {
                Console.Write("\\/");
            }
            Console.WriteLine('-');
        }

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

            PrintHeaderRow(n);
            for (int i = 0; i < n - 2; i++)
            {
                PrintMiddleRow(n);
            }
            PrintHeaderRow(n);
        }
    }
}
```

Задача 4.5. Конвертор за температури

Създайте метод, който конвертира температура от Фаренхайт в Целзий. Форматирайте резултата до втория десетичен знак.

Използвайте формулата: (fahrenheit - 32) * 5 / 9.

Примери

Вход	Изход
95	35.00
33.8	1.00
-40	-40.00

Подсказки

1. Прочетете входните данни
2. Създайте метод, който връща стойност от тип **double**:
3. Извикайте метода в main и запишете върнатата стойност в нова променлива:

Решение

```
namespace TemperaturesConverter
{
    class Program
    {
        static double FahrenheitToCelsius(double degrees)
        {
            double celsius = (degrees - 32) * 5 / 9;
            return celsius;
        }
```

```
        static void Main(string[] args)
        {
            double fahrenheit = double.Parse(Console.ReadLine());
            double celsius = FahrenheitToCelsius(fahrenheit);
            Console.WriteLine("{0:F2}", celsius);
        }
    }
}
```

Задача 4.6. Пресмятане на лице на триъгълник

Създайте метод, който изчислява и връща лицето на триъгълник по дадени основа и височина:

Пример

Вход	Изход
3 4	6

Подсказки

1. Първо прочетете входните данни
2. После създайте метод, но този път вместо да пишете "static void" преди името му, напишете "static double", така че да го накараме да върне стойност от тип double:
3. Извикайте метода в main и съхранете върнатата стойност в нова променлива:

Решение

```
namespace TriangleSurface
{
    class Program
    {
        static double GetTriangleArea(double width, double height)
        {
            return width * height / 2;
        }

        static void Main(string[] args)
        {
            double width = double.Parse(Console.ReadLine());
            double height = double.Parse(Console.ReadLine());
            double area = GetTriangleArea(width, height);
            Console.WriteLine(area);
        }
    }
}
```

Задача 4.7. Повдигане на степен

Създайте метод, който пресмята и връща стойността на число, повдигнато на указаната степен:

Примери

Вход	Изход
2 8	256
3 4	81

Подсказки

1. Както обикновено, прочетете входните данни
2. Създайте метод, който ще има два параметъра - числото и степента, и ще връща резултат от тип double:
3. Отпечатайте резултата

Решение

```
namespace Solution
{
    class Program
    {
        static double RaiseToPower(double number, int power)
        {
            double result = 1d;
            for (int i = 0; i < power; i++)
            {
                result *= number;
            }
            return result;
        }

        static void Main(string[] args)
        {
            double number = double.Parse(Console.ReadLine());
            int power = int.Parse(Console.ReadLine());
            double result = RaiseToPower(number, power);
            Console.WriteLine(result);
        }
    }
}
```

Задача 4.8. По-голямата от две стойности

Имате подадени като входни данни две стойности от един и същи тип. Стойностите може да са от тип int, char или string. Създайте метод GetMax() който връща по-голямата от двете стойности:

Примери

Вход	Изход
int 2	16

16	
char a z	z
string Ivan Todor	Todor

Подсказки

1. За тази задача ще трябва да създадете три метода с едно и също име и с различни сигнатури
2. Създайте метод, който ще сравнява цели числа:
3. Създайте втори метод със същото име, който ще сравнява символи. Следвайте логиката на предния метод:
4. И накрая създайте метод за сравняване на низове. Той ще е малко по-различен, тъй като низовете не може да бъдат сравнявани с операторите > и < Трябва да използвате метода "CompareTo()", който връща целочислена стойност (положителна ако сравняваният обект е по-голям, отрицателна, ако е по-малък и нула, ако двата са равни).
5. Последната стъпка е да прочетете входните данни, да използвате променливи от подходящ тип и да извикате GetMax() от вашия Main():

Решение

```
namespace MaxOfTwoValues
{
    class Program
    {
        static int GetMax(int first, int second)
        {
            if (first >= second)
            {
                return first;
            }
            else
            {
                return second;
            }
        }

        static char GetMax(char first, char second)
        {
            if (first >= second)
            {
                return first;
            }
            else
            {
                return second;
```

```
            }
        }

        static string GetMax(string first, string second)
        {
            if (first.CompareTo(second) >= 0)
            {
                return first;
            }
            else
            {
                return second;
            }
        }

        static void Main(string[] args)
        {
            var type = Console.ReadLine();
            switch (type)
            {
                case "int":
                    {
                        int first = int.Parse(Console.ReadLine());
                        int second = int.Parse(Console.ReadLine());
                        int max = GetMax(first, second);
                        Console.WriteLine(max);
                        break;
                    }

                case "char":
                    {
                        char first = char.Parse(Console.ReadLine());
                        char second = char.Parse(Console.ReadLine());
                        char max = GetMax(first, second);
                        Console.WriteLine(max);
                        break;
                    }

                case "string":
                    {
                        string first = Console.ReadLine();
                        string second = Console.ReadLine();
                        string max = GetMax(first, second);
                        Console.WriteLine(max);
                        break;
                    }
            }
        }
    }
}
```

Задача 4.9. Умножаване на четни по нечетни

Създайте програма, която прочита цяло число и умножава сумата на всичките му нечетни цифри по сумата на всичките му четни цифри:

Примери

Вход	Изход	Коментари
12345	54	12345 има 2 четни цифри - 2 и 4. Сумата им е 6. Също така числото има 3 нечетни цифри - 1, 3 и 5. Сумата им е 9. Умножаваме 6 по 9 и получаваме 54.
-12345	54	

Подсказки

1. Създайте метод с име, описващо предназначението му (например GetMultipleOfEvensAndOdds). Той трябва да има един целочислен параметър и целочислена връщана стойност. Този метод ще извиква два други метода:
2. Създайте два други метода, всеки от които ще сумира четните или нечетните цифри
3. Опишете логиката за сумиране на четни цифри:
4. Направете същото за метода, който ще сумира нечетните цифри
5. Като тествате решението, може да забележите, че то не работи за отрицателни числа. Проследете изпълнението на програмата стъпка по стъпка и поправете грешката (идейка: може да използвате Math.Abs())

Решение

```
namespace OddAndEvenMultiplication
{
    class Program
    {
        private static int GetSumOfEvenDigits(int n)
        {
            n = Math.Abs(n); // fix
            int sum = 0;
            while (n > 0)
            {
                int lastDigit = n % 10;
                if (lastDigit % 2 == 0)
                {
                    sum = sum + lastDigit;
                }
                n /= 10;
            }
            return sum;
        }

        private static int GetSumOfOddDigits(int n)
        {
            n = Math.Abs(n); // fix
            int sum = 0;
            while (n > 0)
            {
                int lastDigit = n % 10;
```

```
                if (lastDigit % 2 != 0)
                {
                    sum = sum + lastDigit;
                }
                n /= 10;
            }
            return sum;
        }

        private static int GetMultiplePfEvensAndOdds(int n)
        {
            int sumEvens = GetSumOfEvenDigits(n);
            int sumOdds = GetSumOfOddDigits(n);
            return sumEvens * sumOdds;
        }

        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            int product = GetMultiplePfEvensAndOdds(n);
            Console.WriteLine(product);
        }
    }
}
```

Задача 4.10. Дебъгване на кода: Почивни дни между две дати

Имате задачата да откриете и поправите грешките във вече написан програмен код, като използвате дебъгера на Visual Studio. За целта трябва да проследите изпълнението на програмата, за да откриете тези редове от кода ѝ, които пораждат неправилен или неочакван резултат.

Разполагате с програма (т.е. със съществуващ програмен код) който се опитва да преброи неработните дни между две дати подадени във формат ден.месец.година (например между 1.05.2015 и 15.05.2015 има 5 неработни дни - събота и недели).

Примери

Вход	Изход	Коментари
1.05.2016 15.05.2016	5	Има 5 неработни дни (събота и недели) в този период: 1-May-2016, 7-May-2016, 8-May-2016, 14-May-2016, 15-May-2016
1.5.2016 2.5.2016	1	Само 1 неработен ден: 1.05.2016 (неделя)
15.5.2020 10.5.2020	0	Втората дата е преди първата. Няма дати в този диапазон.

22.2.2020 1.3.2020	4	По две съботи и недели: • **22.02.2020 и 23.02.2020** • **29.02.2020 и 1.03.2020**

Можете да намерите неработещия програмен код в платформата, файл: Broken-Solutions-1.zip. Изглежда така:

HolidaysBetweenTwoDates.cs

```
using System;
using System.Globalization;

class HolidaysBetweenTwoDates
{
    static void Main()
    {
        var startDate = DateTime.ParseExact(Console.ReadLine(),
            "dd.m.yyyy", CultureInfo.InvariantCulture);
        var endDate = DateTime.ParseExact(Console.ReadLine(),
            "dd.m.yyyy", CultureInfo.InvariantCulture);
        var holidaysCount = 0;
        for (var date = startDate; date <= endDate; date.AddDays(1))
            if (date.DayOfWeek == DayOfWeek.Saturday &&
                date.DayOfWeek == DayOfWeek.Sunday) holidaysCount++;
        Console.WriteLine(holidaysCount);
    }
}
```

Подсказки

Има 4 грешки в кода. Трябва да използвате дебъгера за да ги откриете и поправите. След като сте готови, изпратете поправеният от вас код в платформата.

Решение

```
using System.Globalization;

namespace Solution
{
    class HolidaysBetweenTwoDates
    {
        static void Main()
        {
            var startDate = DateTime.ParseExact(Console.ReadLine(),
                "d.M.yyyy", CultureInfo.InvariantCulture);

            var endDate = DateTime.ParseExact(Console.ReadLine(),
                "d.M.yyyy", CultureInfo.InvariantCulture);
```

```
            var holidaysCount = 0;

            for (var date = startDate; date <= endDate; date = date.AddDays(1))
            {
                if (date.DayOfWeek == DayOfWeek.Saturday ||
                    date.DayOfWeek == DayOfWeek.Sunday)
                {
                    holidaysCount++;
                }
            }

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

Задача 4.11. Price Change Alert

Имате за задача да преработите готов код, който работи без грешки, но не е форматиран както трябва.

Предоставената ви програма следи цените на стоки и дава информация за значимостта на всяка промяна в цената. Според това, доколко е съществена, има четири типа промени: без промяна (цената е същата като предишната), малка (разлика под прага на значимост), цената расте и цената намалява.

Можете да намерите кода за поправяне във файла Broken-Solutions-1.zip.

Вход

- На първия ред получавате N - броят на цените
- На втория ред получавате прага на значимост
- На следващите N реда, получавате цените

Изход

- Не отпечатвате нищо за първата цена
- Ако няма разлика от предишната цена, съобщението е: "NO CHANGE: {текуща цена}"
- При малка разлика: "MINOR CHANGE: {предишна цена} to {текуща цена} ({разлика}%)"
- При голяма разлика: "PRICE UP: {предишна цена} to {текуща цена} ({разлика}%)" or "PRICE DOWN: {предишна цена} to {текуща цена} ({разлика}%)"

Процентите трябва да са закръглени до втория знак след десетичната точка.

Примери

Вход	Изход

3 0.1 10 11 12	PRICE UP: 10 to 11 (10.00%) MINOR CHANGE: 11 to 12 (9.09%)
3 0.1 10 10 12	NO CHANGE: 10 PRICE UP: 10 to 12 (20.00%)

Подсказки

1. Изтеглете програмния код и се запознайте с него
2. Заемете се първо с лошото форматиране на кода - махнете ненужните празни редове, вмъкнете навътре кода където и колкото е нужно
3. Коригирайте имената на параметрите на методите
4. Дайте и на методите подходящи имена

Решение

```
namespace PriceChangeAlert
{
    class Program
    {
        private static string GetMessage(double currentPrice, double lastPrice,
double priceDiff, bool isSignificantDifference)
        {
            string message = "";
            if (priceDiff == 0)
            {
                message = string.Format("NO CHANGE: {0}", currentPrice);
            }
            else if (!isSignificantDifference)
            {
                message = string.Format("MINOR CHANGE: {0} to {1} ({2:p2})",
lastPrice, currentPrice, priceDiff);
            }
            else if (isSignificantDifference && (priceDiff > 0))
            {
                message = string.Format("PRICE UP: {0} to {1} ({2:p2})", lastPrice,
currentPrice, priceDiff);
            }
            else if (isSignificantDifference && (priceDiff < 0))
                message = string.Format("PRICE DOWN: {0} to {1} ({2:p2})",
lastPrice, currentPrice, priceDiff);
            return message;
        }

        private static bool SignificantDifference(double limit, double isDiff)
        {
            if (Math.Abs(limit) >= isDiff)
```

```
            {
                return true;
            }
            return false;
        }

        private static double CalculatePriceDifference(double last, double price)
        {
            return (price - last) / last;
        }

        static void Main()
        {
            int n = int.Parse(Console.ReadLine());
            double limit = double.Parse(Console.ReadLine());
            double lastPrice = double.Parse(Console.ReadLine());

            for (int i = 0; i < n - 1; i++)
            {
                double ciurrentPrice = double.Parse(Console.ReadLine());
                double priceDiff = CalculatePriceDifference(lastPrice,
ciurrentPrice);
                bool isSignificantDifference = SignificantDifference(priceDiff,
limit);

                string message = GetMessage(ciurrentPrice, lastPrice, priceDiff,
isSignificantDifference);
                Console.WriteLine(message);

                lastPrice = ciurrentPrice;
            }
        }
    }
}
```

Тема 5. Символни низове

Задача 5.1. Преобразуване от 10-ична в N-ична ПБС

Напишете програма, която получава число в в 10-ична бройна система и го преобразува в число в N-ична бройна система, където 2 < = N < = 10. Входът се състои от 1 ред, съдържащ две числа, разделени с един интервал. Първото число е основа N, към която трябва да преобразувате. Вторият е число в 10-ична бройна система. Не използвайте никакви вградени функционалности за преобразуване на числа, опитайте се да напишете свой собствен алгоритъм.

Упътване

За алгоритъм (от 10-ична в 2-ична) можете да прочетете тази статия.

Алгоритъмът за преобразуване на число от 10-ична в 2-ична бройна система е подобен: вместо "% 2", ползвайте "% N".

Вход

На един ред въвеждате основа на бройната система и число в 10-ична бройна система

Изход

На един ред извеждате числото в N-ична бройна система

Примери

Вход	Изход
7 10	13
3 154	12201
5 123	443
4 1000	33220
9 3487	4704

Решение

```
namespace Convert10ToN
{
    class Program
    {
        static string Convert(int number, int target)
        {
            string result = String.Empty;
            while (number > 0)
            {
                var remainder = number % target;
                result = string.Concat(remainder.ToString(), result);
                number = number / target;
            }
            return result;
        }

        static void Main(string[] args)
        {
            var line = Console.ReadLine().Split().Select(int.Parse).ToArray();
            int target = line[0];
            int number = line[1];
            Console.WriteLine(Convert(number, target));
        }
    }
}
```

Задача 5.2. Преобразуване от N-ична в 10-ична ПБС

Напишете програма, която взема N-ично число и го преобразува 10-ично число (0 до 1050), където от 2 < = N < = 10. Входът се състои от 1 ред, съдържащ две числа, разделени с един интервал. Първото число е основата N, към която трябва да преобразувате. Второто е числото N, което трябва да се преобразува. Не използвайте никакви вградена функционалности за преобразуване, опитайте се да напишете свой собствен алгоритъм

Вход

На един ред въвеждате основа на бройната система и число в N-ична бройна система

Изход

На един ред извеждате числото в 10-ична бройна система

Упътване

Вижте тази картина за повече яснота за преобразуване от 2-ична в 10-ична БС. Отново, алгоритъмът за преобразуване от N-ична БС е подобен.

Примери

Вход	Изход
7 13	10
3 12201	154
5 443	123
4 33220	1000
9 4704	3487

Решение

```
namespace ConvertNto10
{
    class Program
    {
        static string Convert(int number, int target)
        {
            double sum = 0, pow = 0;
            while (number > 0)
            {
                var current = number % 10;
                sum += current * Math.Pow(target, pow);
                pow++;
                number /= 10;
            }
            return sum.ToString();
        }

        static void Main(string[] args)
        {
            var line = Console.ReadLine().Split().Select(int.Parse).ToArray();
            int target = line[0];
            int number = line[1];
            Console.WriteLine(Convert(number, target));
        }
    }
}
```

Задача 5.3. Обръщане на низ

Създайте метод, който получава низ и връща низ, получен от същите символи, но в обратен ред.

Вход

На един ред получавате символен низ

Изход

На един ред извеждате обърнатия низ

Ограничения

Символният низ да се състои от една дума, т.е. да няма интервали и да не се ползва метода Reverse

Упътване

Може да отпечатате всички символи на низа, като го обходите отзад напред или да конструирате нов низ, в който да прехвърлите символите на първия, в обратен ред

Примери

Вход	Изход
a	a
aba	aba
alenafanela	alenafanela
alibaba	ababila
baba	abab

Решение

```
namespace StringReverse
{
    class Program
    {
        static string StringReverse(string input)
        {
            string result = String.Empty;
            for (int i = input.Length - 1; i >= 0; i--)
            {
                result = String.Concat(result, input[i].ToString());
            }
            return result;
        }

        static void Main(string[] args)
        {
            var input = Console.ReadLine();
            Console.WriteLine(StringReverse(input));
        }
    }
```

```
}
```

Задача 5.4. Unicode Символи

Напишете програма, която преобразува символен низ в последователност от Unicode символни *кодове.*

Вход

На един ред въвеждате символен низ

Изход

На един ред извеждате Unicode на всеки символ

Примери

Вход	Изход
Hi!	\u0048\u0069\u0021
What?!?	\0057\0068\0061\0074\003f\0021\003f

Решение

```
namespace TextToUnicode
{
    class Program
    {
        static string TextToUnicode(string input)
        {
            string result = String.Empty;
            for (int i = 0; i < input.Length; i++)
            {
                string code = "\\" + ((int)input[i]).ToString("X4");
                result = String.Concat(result, code.ToLower());
            }
            return result;
        }

        static void Main(string[] args)
        {
            var text = Console.ReadLine();
            var unicode = TextToUnicode(text);
            Console.WriteLine(unicode);
        }
    }
}
```

Задача 5.5. Умножаване на символни кодове

Създайте метод, който получава два низа като аргументи и връща сбора от техните произведения от символни кодове на съответни позиции (умножете str1.charAt (0) с str2.charAt (0) и ги добавете към сбора). След това продължете със следващите два знака. Ако един от низовете е по-дълъг от другия, добавете останалите символни кодове към сбора без умножение.

Вход

На един ред въвеждате два низа

Изход

На един ред извеждате сбора от техните произведения от символни кодове на съответни позиции

Примери

Вход	Изход
Gosho Pesho	53253
123 522	7647
a aaaa	9700

Решение

```
namespace CharMultiplication
{
    class Program
    {
        static int CharMultiplication(string word1, string word2)
        {
            word1 = new string(word1.Reverse().ToArray());
            word2 = new string(word2.Reverse().ToArray());
            int shortest = Math.Min(word1.Length, word2.Length) - 1;

            string word3 = String.Empty;
            if (word1.Length > word2.Length) word3 = word1.Substring(shortest,
word1.Length - 1);
            if (word1.Length < word2.Length) word3 = word2.Substring(shortest,
word2.Length - 1);

            int sum = 0;
            for (int i = 0; i <= shortest; i++)
            {
                sum += (int)word1[i] * (int)word2[i];
            }
            for (int i = 0; i < word3.Length; i++)
            {
                sum += (int)word3[i];
            }

            return sum;
        }

        static void Main(string[] args)
        {
            var line = Console.ReadLine().Split().ToArray();
            var word1 = line[0];
            var word2 = line[1];
            Console.WriteLine(CharMultiplication(word1, word2));
        }
    }
}
```

Задача 5.6. Палиндром

Създайте метод, който получава низ и връща True или False в зависимост от това дали думата е палиндром или не

Вход

На един ред поучавате символен низ

Изход

На един ред извеждате True, ако низа е палиндром или False, ако не е.

Ограничения

Символният низ да се състои от една дума, т.е. да няма интервали

Упътване

Един низ е палиндром, ако прочетен отзад напред е същия, какъвто е и когато го четем отпред назад

Примери

Вход	Изход
a	True
aba	True
alenafanela	True
alibaba	False
baba	False

Решение

```
namespace Palindrom
{
    class Program
    {
        static bool Palindrom(string word)
        {
            string reversed = new string(word.Reverse().ToArray());
            if (word.CompareTo(reversed) == 0) return true;
            else return false;
        }

        static void Main(string[] args)
        {
            string word = Console.ReadLine();
            Console.WriteLine(Palindrom(word));
        }
    }
}
```

Задача 5.7. Магически променящи се думи

Напишете метод, който приема като вход два низа и връща True или False, ако те са заменяеми, или не. Заменяеми са думи, където символите в първия

низ може да бъдат заменени и да се получи втория низ. Пример: "egg" и "add" са заменяеми, но "aabbccbb" и "nnooprzz" не са. (Първото "b" отговаря на "o", но тогава то също така отговаря на"z"). Двете думи може да нямат една и съща дължина, ако случаят е такъв, те са заменяеми, само ако по-дългата няма повече от видовете букви на първата ("Clint" и "Eastwaat" са заменяеми защото "a" и "t" вече са заменени като "l" и " n " но "Clint" и "Eastwood" не са заменяеми защото 'o' и ,'d' не се съдържат в"Clint").

Примери

Вход	Изход
gosho hapka	true
aabbaa ddeedd	true
foo bar	false
Clint Eastwood	false

Решение

```
namespace MagicallyChangingWords
{
    class Program
    {
        static bool IsReplacable(char[] letters1, char[] letters2)
        {
            var matches = new Dictionary<char, char>();
            var smallerArr = letters1.Length == (Math.Min(letters1.Length,
letters2.Length)) ? letters1 : letters2;
            var biggerArr = letters1.Length == (Math.Min(letters1.Length,
letters2.Length)) ? letters2 : letters1;

            for (int i = 0; i < smallerArr.Length; i++)
            {
                if (!matches.ContainsKey(smallerArr[i]))
                {
                    matches.Add(smallerArr[i], biggerArr[i]);
                }
                else
                {
                    if (matches[smallerArr[i]] == biggerArr[i])
                    {
                        continue;
                    }
                    else
                    {
                        return false;
                    }
                }
            }

            for (int i = smallerArr.Length; i < biggerArr.Length; i++)
            {
                if (matches.ContainsValue(biggerArr[i]))
```

```
                {
                    continue;
                }
                else
                {
                    return false;
                }
            }
            return true;
        }

        static void Main(string[] args)
        {
            var input = Console.ReadLine().Split().ToArray();
            var word1 = input[0].ToArray();
            var word2 = input[1].ToArray();
            Console.WriteLine(IsReplacable(word1, word2));
        }
    }
}
```

Задача 5.8. Сбор на големи числа

Входните данни са два реда – въвеждат се две числа, които може да са големи (от 0 до 1050). Трябва да изведете сбора на тези числа.

Забележка: не използвайте BigInteger или BigDecimal класове за решаването на този проблем.

Примери

Вход	Изход
23 23	46

Вход	Изход
9999 1	10000

Input	Output
923847238931983192462832102 934572893617836459843471846187346	934573817465075391826664309019448

Решение

```
namespace BigNumbersSum
{
    class Program
    {
        static String Sum(String A, String B)
        {
            String C = String.Empty;

            if (A.Length < B.Length) A = new String('0', B.Length - A.Length) + A;
            else B = new String('0', A.Length - B.Length) + B;

            int j = 0, PART = 0;
```

```
            for (int i = A.Length - 1; i >= 0; i--)
            {
                int SUM = (int)A[i] + (int)B[i] - 96; // ASCII
                if (PART > 0)
                {
                    SUM += PART;
                    PART = 0;
                }
                if (SUM > 9)
                {
                    PART = SUM / 10;
                    SUM = SUM % 10;
                }
                C += SUM.ToString();
                j++;
            }

            return string.Join("", C.Reverse());
        }

        static void Main(string[] args)
        {
            String A = Console.ReadLine();
            String B = Console.ReadLine();
            Console.Write(Sum(A, B));
        }
    }
}
```

Задача 5.9. Умножаване на големи числа

Входните данни са два реда – на първия се въвежда голямо число (от 0 до 1050). На втория - едноцифрено число (0-9). Трябва да се изведе произведението на тези числа. Забележка: не използвайте класовете BigInteger или BigDecimal за решаването на този проблем.

Примери

Вход	Изход
23 2	46

Вход	Изход
9999 9	89991

Вход	Изход
923847238931983192 462832102 4	934573817465075391 826664309019448

Решение

```
namespace BigNumbersMultiplication
{
    class Program
    {
        static string Multiply(string BigNumber, int Multiplyer)
        {
            String Result = String.Empty;
            int j = 0, Reminder = 0;
            for (int i = BigNumber.Length - 1; i >= 0; i--)
            {
                int Current = Multiplyer * ((int)BigNumber[i] - 48); // ASCII
```

```
                if (Reminder > 0)
                {
                    Current += Reminder;
                    Reminder = 0;
                }
                if (Current > 9)
                {
                    Reminder = Current / 10;
                    Current = Current % 10;
                }
                Result += Current.ToString();
                j++;
            }

            if (Reminder > 0) Result += string.Join("",
Reminder.ToString().Reverse());

            return string.Join("", Result.Reverse());
        }

        static void Main(string[] args)
        {
            string BigNumber = Console.ReadLine();
            int Multiplyer = int.Parse(Console.ReadLine());
            Console.WriteLine(Multiply(BigNumber, Multiplyer));
        }
    }
}
```

Задача 5.10. Обработка на числа с представки и наставки

Наков обича математиката. Но той също се интересува от английската азбука много. Той е изобретил игра с цифри и букви от английската азбука. Играта е проста. Получавате низ, състоящ се от число между две букви. В зависимост от това дали буквата е пред числото или след него ще извършвате различни математически операции с числото за постигане на резултат.

Първо започнете с буквата преди числото.

- Ако тя е главна, делите на позицията на буквата в азбуката.
- Ако тя е малка, умножавате числото по позицията на буквата в азбуката.

После преминаваш към буквата след числото.

- Ако тя е главна изваждате позицията си от полученото число.
- Ако тя е малка добавяте позицията си към полученото число.

Но играта става твърде лесно за Наков и наистина се справя бързо. Той решава да я усложни малко, като правилата са същите, но с множество низове, като се иска общата сума на всички резултати от стринговете. След като той започна да решава задачата с повече низове и по-големи числа, ставаше доста трудно да смята наум. Така той любезно ви моли да

напишете програма, която изчислява сумата на всички числа, след извършените операции на всяко число.

Например

Дадена е последователността "A12b s17G": имаме два низа -"A12b" и "s17G". Извършваме операциите на всяко от числата и ги събираме. Започваме с буквата преди числото на първия низ. A е главна и позицията в азбуката е 1. Така че разделяме числото 12 на позиция 1 (12/1 = 12). Тогава минаваме към буквата след числото. b е малка и неговата позиция е 2. Така че ние добавяме 2 към полученото число (12 + 2 = 14). По същия начин за втория низ s е малка и нейната позиция е 19, така че ние умножаваме числото (17 * 19 = 323). Тогава ние имаме главна буква G с позиция 7, така че ние изваждаме от резултата 7 (323 – 7 = 316). И накрая ние събираме 2 резултата и получаваме 14 + 316 = 330.

Вход

- Входът е на един ред, съдържащ последователност от символни низове. Низовете са разделени от един или повече интервали.
- Входните данни винаги ще бъде валидни и в описания формат. Няма нужда да го проверите изрично.

Изход

- Печат на конзолата на едно число: общата сума от всички обработени числа, закръглени до две цифри след десетичния разделител

Ограничения

- The count of the strings will be in the range [1 … 10].
- The numbers between the letters will be integers in range [1 … 2 147 483 647].
- Time limit: 0.3 sec. Memory limit: 16 MB.
- Броят на низовете ще бъдат в интервала [1… 10].
- Числата между буквите ще бъде цели числа в диапазона [1… 2 147 483 647].
- Време: до 0,3 сек, памет : до 16 MB.

Примери

Вход	Изход	Коментари
A12b s17G	330.00	12/1=12, 12+2=14, 17*19=323, 323-7=316, 14+316=330
P34562Z q2576f H456z	46015.13	
a1A	0.00	

Решение

```
namespace PrefixesAndSuffixes
{
    class Program
```

```
    {
        static void Main(string[] args)
        {
            var input = Console.ReadLine().Split(' ');
            var sum = 0d;
            foreach (var a in input)
            {
                var b = double.Parse(a.Substring(1, a.Length - 2));
                var l1 = Char.IsLower(a.First());
                var l2 = Char.IsLower(a.Last());

                b *= (l1 ? a.ToLower().First() - 96 : 1 / (a.ToLower().First() -
96));
                b += (l2 ? a.ToLower().Last() - 96 : -(a.ToLower().Last() - 96));

                sum += b;
            }
            Console.WriteLine("{0:f2}", sum);
        }
    }
}
```

Задача 5.11. Разклащане на Мелрах

Даден е низ от случайни символи, както и шаблон от случайни символи. Вие трябва да "отърсите" (премахнете) всички граничните случаи на този шаблон, с други думи, първото съвпадение (срещане) и последното съвпадение на шаблона в низа.

Когато успешно премахнете дадено съвпадение , премахнете от шаблона символа, който съответства на индекса, равен на дължината на шаблона / 2. След това продължавате да премахвате от краищата на низа съвпаденията с новия шаблон, докато шаблона стане празен или докато основния низ стане по-къс от шаблона.

В случай, че сте намерили най-малко две съвпадения, успешно сте ги премахнали , извеждате "Shaked it" на конзолата. В противен случай извеждате "No shake.", остатъкът от основния низ, и завършвате програмата. Вижте примерите за повече информация.

Вход

- Входът се състои от два реда.
- На първия ред вие ще получите низ от случайни символи.
- На втори ред вие ще получите шаблон.

Изход

- Извеждате "`Shaked it.`" винаги, при успешно разклащане на Мелрах.
- Ако разклащането на Мелрах не успее, извеждате "`No shake.`" и на следващия ред извеждате какво е останало от главния символен низ.

Ограничения

- Двата низа може да съдържат всякакви ASCII символи.

- Позволеното време/памет: 250ms/16 MB.

Примери

Вход	Изход
astalavista baby sta	Shaked it. No shake. alavi baby

Вход	Изход
##mtm!!mm.mm*mtm.# mtm	Shaked it. Shaked it. No shake. ##!!.*.#

Решение

```
namespace MelrahShaking
{
    class Program
    {
        static void Main(string[] args)
        {
            var text = Console.ReadLine();
            var template = Console.ReadLine();

            while (true)
            {
                int counter = 0;

                while (text.Contains(template))
                {
                    text = text.Remove(text.IndexOf(template), template.Length);
                    counter++;
                }

                if (counter == 0)
                {
                    Console.WriteLine("No shake.");
                    break;
                }

                if (counter > 1) Console.WriteLine("Shake it.");

                template = template.Remove(template.Length / 2, 1);
            }
            Console.WriteLine(text);
        }
    }
}
```

Задача 5.12. Само букви

Напишете програма, която въвежда низ съобщение като вход и замества всички числа с буквата непосредствено след числото.

Вход

На един ред се въвежда съобщение, което трябва да се поправи

Изход

Изведете само поправеното съобщение.

Примери

Вход	Изход
ChangeThis12andThis56k	ChangeThisaandThiskk

Вход	Изход
1Beware72ForThe4End88888	BBewareFForTheEEnd88888

Решение

```
namespace OnlyLetters
{
    class Program
    {
        static void Main(string[] args)
        {
            string line = Console.ReadLine();
            string last = "", result = "";
            for (int i = line.Length - 1; i >= 0; i--)
            {
                if (char.IsDigit(line[i]))
                {
                    if (last != "") result += last;
                    else result += line[i].ToString();
                }
                else
                {
                    last = line[i].ToString();
                    result += line[i].ToString();
                }
            }
            Console.WriteLine(new string(result.Reverse().ToArray()));
        }
    }
}
```

Задача 5.13. Скривалището

Вие сте детектив от Скотланд Ярд и трябва да намерите скривалището на много опасна група от престъпници. Вие ще получите карта под формата на низ и след това ще получите улики от разузнаването. На следващите неизвестен брой редове ще получавате масиви съдържащи два елемента:

- Първият елемент ще бъде символ, който бележи скривалището.
- Вторият елемент ще бъде минимален брой символи, които трябва да търсите.

Масивът ще бъде във формат: "**{searchedCharacter} {minimumCount}**".

Ако не можете да намерите скривалище ➔ продължете да четете следващите два реда

Ако откриете скривалище ➔ прекъсвате програмата и извеждате индекс, където започва скривалището и дължината на скривалището.

Вход

- На първия ред ще получите карта, която ще съдържа случайни низове.
- На следващата неизвестен брой редове ще получите масиви
  - Първият елемент е търсения символ
  - Вторият елемент е минималният брой, които трябва да се търси

Изход

Ако откриете скривалището, печат:

"Hideout found at index {indexOfTheFirstChar} and it is with size {lengthOfTheFoundString}!"

Примери

Вход	Изход
asd@@asdasd@@@@@@@asdasd asdsad @ 5	Hideout found at index 11 and it is with size 7!

Вход	Изход
asd@@asd***asdasdsad123%4521Asd sad**********ASssda & 3 * 20 * 10 * 2	Hideout found at index 34 and it is with size 12!

Решение

```
namespace Hideout
{
    class Program
    {
        static void Main(string[] args)
        {
            var text = Console.ReadLine();
            while (true)
            {
```

```
            var line = Console.ReadLine().Split().ToArray();
            var ch = char.Parse(line[0]);
            var count = int.Parse(line[1]);

            if (text.Contains(new string(ch, count)))
            {
                var firstCharIndex = text.IndexOf(new string(ch, count));
                var stringLength = count;

                for (int i = firstCharIndex + count; i < text.Length; i++)
                {
                    if (text[i] == ch)
                    {
                        stringLength++;
                    }
                }

                Console.WriteLine($"Hideout found at index {firstCharIndex} and
it is with size {stringLength}!");
            }
        }
    }
}
```

Задача 5.14. Цензура

Напишете програма, която приема като входни данни, една дума и изречение. Вашата програма трябва да търси думата в изречението и замени всяка буква от думата с "*". Вие трябва да направите това за всяко срещане на думата. Заменете само думите, които са напълно еднакви с думата на първия ред. Обърнете внимание, че трябва да се замени думата, дори ако тя е част от друга дума.

Вход

Входът ще се състои от два реда:

- На първия ред, ще бъде дума, която трябва да се цензурира.
- На втори ред ще бъдат изречението, които трябва да се цензурира.

Изход

Отпечатате изречението, след цензурирането.

Примери

Вход	Изход
money **Show me the money**	**Show me the *****
Doom **Doom and Gloom**	****** and Gloom**

Java I love Java and JavaScript, but I hate Rxjava	I love **** and ****Script, but I hate Rxjava

Решение

```
namespace Censorship
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] banWords = Console.ReadLine().Split(' ');
            string text = Console.ReadLine();

            foreach (var banWord in banWords)
            {
                if (text.Contains(banWord))
                {
                    text = text.Replace(banWord, new string('*', banWord.Length));
                }
            }

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

Задача 5.15. StringBuilder

Напишете програма, която приема като входни данни на първи ред символен низ. На следващия ред получава следните команди в такъв формат:

```
Append <Символен низ>
Remove  <pos> <number>
Insert <pos> <string>
Replace <substring> <substring>
```

Накрая извежда резултатния символен низ.

Вход

Входът ще се състои от два реда:

- **На първия ред, ще бъде символния низ, който ще се променя**
- **На втори ред ще бъде въведена команда с параметрите в указания формат**

Изход

Отпечатате изречението, след цензурирането.

Примери

Вход	Изход

That is not true. Remove 8 3	That is true.
That is integer Insert 7 Big	That is Big integer
That type is string Replace string int	That type is int

Решение

```
namespace StringBuilder
{
    class Program
    {
        static void Main(string[] args)
        {
            var input = Console.ReadLine();
            var cmd = Console.ReadLine().Split().ToArray();

            var sb = new System.Text.StringBuilder();
            sb.Append(input);

            switch (cmd[0])
            {
                case "Append":
                    sb.Append(cmd[1]);
                    break;
                case "Remove":
                    {
                        var pos = int.Parse(cmd[1]);
                        var number = int.Parse(cmd[2]);
                        sb.Remove(pos, number);
                        break;
                    }
                case "Insert":
                    {
                        var pos = int.Parse(cmd[1]);
                        var text = int.Parse(cmd[2]);
                        sb.Insert(pos, text);
                        break;
                    }
                case "Replace":
                    sb.Replace(cmd[1], cmd[2]);
                    break;
            }

            Console.WriteLine(sb.ToString());
        }
    }
}
```

Задача 5.16. Изпрати ми Email

Снощи Пешо е получил имейл от едно момиче. За съжаление той не може да си спомни дали тя си струва. Той има план за това как да решите дали той

трябва да напише съобщение на момичето и се нуждае от умения за програмиране. Той ще ви даде мейла си и вашата задача е да се извади сборът от символите след "@" от сбора на символите преди ' @'. Ако резултатът е равен или по-голямо от 0 - той ще напише мейла си, иначе не.

Вход

Вие ще получите един ред с имейл на момичето.

Изход

Ако резултатът е равен или по-голямо от 0 извежда:

- Call her!

Иначе извежда:

- She is not the one.

Примери

Вход	Изход
maria@abv.bg	She is not the one.
gergana.ivanova@yahoo.com	Call her!

Решение

```
namespace SendMeMail
{
    class Program
    {
        static void Main(string[] args)
        {
            var email = Console.ReadLine().Split('@').ToArray();
            var part1 = email[0];
            var part2 = email[1];

            int sum1 = 0, sum2 = 0;
            foreach (var item in part1)
            {
                sum1 += (int)item;
            }
            foreach (var item in part2)
            {
                sum2 += (int)item;
            }

            if (sum1 - sum2 >= 0)
            {
                Console.WriteLine("Call her!");
            }
            else
            {
                Console.WriteLine("She is not the one.");
            }
        }
    }
```

```
}
```

Задача 5.17. Изпрати ми Email (Unicode)

Снощи Пешо е получил имейл от едно момиче. За съжаление той не може да си спомни дали тя си струва. Той има план за това как да решите дали той трябва да напише съобщение на момичето и се нуждае от умения за програмиране. Той ще ви даде мейла си и вашата задача е да се извади сборът от Unicode на символите след "@" от сбора на Unicode на символите преди ' @'. Ако резултатът е равен или по-голямо от 0 - той ще напише мейла си, иначе не.

Вход

Вие ще получите един ред с имейл на момичето.

Изход

Ако резултатът е равен или по-голямо от 0 извежда:

- Call her!

Иначе извежда:

- She is not the one.

Примери

Вход	Изход
maria@abv.bg	She is not the one.
gergana.ivanova@yahoo.com	Call her!

Решение

```
namespace SendMeMailUnicode
{
    public class Program
    {
        static void Main(string[] args)
        {
            var email = Console.ReadLine().Split('@').ToArray();

            double part1sum = 0;
            foreach(var item in email[0])
            {
                part1sum += Char.GetNumericValue(item);
            }

            double part2sum = 0;
            foreach (var item in email[1])
            {
                part2sum += Char.GetNumericValue(item);
            }

            if (part2sum - part1sum >= 0)
            {
                Console.WriteLine("Call her!");
```

```
            }
            else
            {
                Console.WriteLine("She is not the one.");
            }
        }
    }
}
```

Задача 5.18. Karate Strings

Петър се опитва да стане карате майстор. Като програмист, Петър има представа как да се обучава, така че той решава да се обучава със символни низове. Ударите му са маркирани с ">". Непосредствено след този символ ще има цяло число, което означава, силата на удара. Трябва да премахнете x символа (където x е силата на удар), започвайки след символа за удар ('> '). Ако намерите друг символ за удар (">") докато изтривате символи, трябва да добавите силата към предишния ви удар. Когато се обработят всички символи, изведете низа без изтритите символи. Не трябва да изтривате символа за удар – '> ", но трябва да изтривате целите числа, които представят силата.

Вход

Вие ще получите един ред със символен низ, който се използва от Пешо за обучение.

Изход

Извежда, което е останало от низа след ударите на Пешо.

Ограничения

- винаги ще получите сила за удари
- низът ще се състои само от букви от латинската азбука, цели числа и символ ">"
- Силата на удара ще бъде в интервала [0... 9]

Примери

Вход	Изход	Коментари
abv>1>1>2>2asdasd	abv>>>>dasd	1ви удар е с индекс 3 и ще е със сила 1. Изтриваме само цифрата след удара. Низът ще изглежда така: abv>>1>2>2asdasd 2ри удар е със сила 1 и символния низ се трансформира до този: abv>>>2>2asdasd 3ти удар е сега със сила 2. Изтриваме цифрата и намираме друг удар. Тук низа изглежда така: abv>>>>2asdasd.

		4ти удар е със сила 2. Имаме сила 1 останала от предишния удар, и добавяме силата на текущия удар към останалото и го добавяме към общата сила, която е 3. Изтриваме следващите 3 символа и получаваме низа abv>>>>dasd Нямаме още удари и извеждаме: abv>>>>dasd

Вход	Изход
pesho>2sis>1a>2akarate>4hexmaster	pesho>is>a>karate>master

Решение

Задача 5.19. Маршрут на робот

Имате робот, който се намира в началото на координатна система Oxy, насочен на дясно (по оста Ox). Той получава команди за движение, които са:

- R: завъртане на 90 градуса на дясно;
- L: завъртане на 90 градуса на ляво;
- F <N>: движение напред <N> метра

От конзолата получавате низ, който представлява последователност от такива команди. Напишете програма, която изчислява позицията на робота след изпълнение на командите и разстоянието на което се е отдалечил от началото на координатната система в следния формат:

- „Position: (x: {xPos}, y:{yPos}), Distance = {Distance}“, разстоянието да се закръгли с *точност 0,01*

Вход

Вие ще получите един ред със символен низ, който представлява маршрута на робота.

Изход

Извежда координатите на позицията му в края на маршрута, както и разстоянието на което се е отдалечил.

Упътване

Използвайте Math.Sqrt (x*x+y*y) за изчисляване ан разстоянието

Ограничения

низът ще се състои само от указаните букви и цели числа

Примери

Вход	Изход
L20R10LL20R10R30	Position: (x: 20, y:30), Distance = 36.06 m

Решение

Задача 5.20. Цензора със StringBuilder

Решете същата задача като ползвате StringBuilder

Напишете програма, която приема като входни данни, една дума и изречение. Вашата програма трябва да търси думата в изречението и замени всяка буква от думата с "*". Вие трябва да направите това за всяко срещане на думата. Заменете само думите, които са напълно еднакви с думата на първия ред. Обърнете внимание, че трябва да се замени думата, дори ако тя е част от друга дума.

Вход

Входът ще се състои от два реда:

- На първия ред, ще бъде дума, която трябва да се цензурира.
- На втори ред ще бъдат изречението, които трябва да се цензурира.

Изход

Отпечатате изречението, след цензурирането.

Примери

Вход	Изход
money Show me the money	Show me the *****
Doom Doom and Gloom	**** and Gloom
Java I love Java and JavaScript, but I hate Rxjava	I love **** and ****Script, but I hate Rxjava

Решение

Тема 6. Многомерни масиви

Задача 6.1. Вход и изход на матрица

Напишете програма, която въвежда брой редове и брой колони. След което въвежда елементите на двумерен масив (матрица) със съответния брой редове и колони. Всички елементи на масива ще са цели числа. Изведете получения двумерен масив

Примери

Вход	Изход	Вход	Изход
2 4 1 2 3 5 8 6 9 4	1 2 3 5 8 6 9 4	3 3 1 2 3 9 8 7 4 5 6	1 2 3 9 8 7 4 5 6

Решение

```
namespace MatrixInputOutput
{
    class Program
    {
        static void Main(string[] args)
        {
            int rows = int.Parse(Console.ReadLine());
            int cols = int.Parse(Console.ReadLine());
            int[,] matrix = new int[rows, cols];

            for (int x = 0; x < rows; x++)
            {
                for (int y = 0; y < cols; y++)
                {
                    matrix[x, y] = int.Parse(Console.ReadLine());
                }
            }

            for (int x = 0; x < rows; x++)
            {
                for (int y = 0; y < cols; y++)
                {
                    Console.Write("{0, 4}", matrix[x, y]);
                }
                Console.WriteLine();
            }
        }
    }
}
```

Задача 6.2. Средноаритметично по редове

Напишете програма, която въвежда брой редове и брой колони. След което въвежда елементите на двумерен масив (матрица) със съответния брой редове и колони. Всички елементи на масива ще са цели числа. Изведете двумерния масив, като на всеки ред прибавите по 1 елемент в края му,

който да бъде равен на средноаритметичното от всички елементи в съответния ред. При извеждане на масива го форматирайте, така че всеки елемент да заема 10 позиции

Примери

Вход	Изход
2 4 1 2 3 5 8 6 9 4	1 2 3 5 2.75 8 6 9 4 6.75

Подсказки

- Когато отпечатвате елемента си, използвайте нещо подобно:
  `Console.Write("{0, 10}", matrix[row, col]);`
- Със започването на всеки ред задавайте стойността на променливата, в която пазите сумата на елементите му на 0. Когато приключвате с обхождането на реда, изчислете средноаритметичното му и го изведете.

Решение

```
namespace ArithmeticMeanOfRows
{
    class Program
    {
        static void Main(string[] args)
        {
            int rows = int.Parse(Console.ReadLine());
            int cols = int.Parse(Console.ReadLine());
            int[,] matrix = new int[rows, cols];

            for (int x = 0; x < rows; x++)
            {
                for (int y = 0; y < cols; y++)
                {
                    matrix[x, y] = int.Parse(Console.ReadLine());
                }
            }

            for (int row = 0; row < rows; row++)
            {
                double avg = 0;
                for (int col = 0; col < cols; col++)
                {
                    Console.Write("{0, 8}", matrix[row, col]);
```

```
                avg += matrix[row, col];
            }
            avg = avg / cols;
            Console.WriteLine("{0, 8}", avg);
        }
    }
}
```

Задача 6.3. Минимум по колони

Напишете програма, която въвежда брой редове и брой колони. След което въвежда елементите на двумерен масив (матрица) със съответния брой редове и колони – елементите на всеки ред от масива ще са на отделен ред. Всички елементи на масива ще са цели числа. Изведете двумерния масив, като накрая добавите един нов ред – всеки елемент в този ред показва минималния елемент от колоната, която стои над него. При извеждане на масива го форматирайте, така че всеки елемент да заема 5 позиции

Примери

Вход	Изход
3 4 1 2 3 5 8 6 9 4 5 8 4 3	1 2 3 5 8 6 9 4 5 8 4 3 1 2 3 3

Подсказки

- Когато въвеждате матрицата, създайте си помощен масив за реда. Въвеждането му от един ред е аналогично на списъците:

```
for(int row = 0; row < rows; row++) {
    int[] rowArray = Console.ReadLine().Split(' ').Select(int.Parse).ToArray();
    for(int col = 0; col < cols; col++) {
        matrix[row, col] = rowArray[col];
    }
}
```

- Създайте едномерен масив, който ще съхранява минималните елементи от всяка колона
- Сменете местата на редовете и колоните в обхождането, така че външния цикъл да отговаря за колоните, а вътрешния за редовете.
- След пълното извъртане на вътрешния цикъл, съхранете намерения минимален елемент в едномерния масив по подобен начин: minElements[col] = min;
- Отпечатайте матрицата, като непосредствено след нея, отпечайте и едномерния масив – използвайте едно и също форматиране.

Решение

```
namespace MinimumByColumns
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            int rows = int.Parse(Console.ReadLine());
            int cols = int.Parse(Console.ReadLine());
            int[,] matrix = new int[rows, cols];

            for (int row = 0; row < rows; row++)
            {
                var line = Console.ReadLine().Split().Select(int.Parse).ToArray();
                for (int col = 0; col < cols; col++)
                {
                    matrix[row, col] = line[col];
                }
            }

            int[] min = new int[cols];
            for (int col = 0; col < cols; col++)
            {
                min[col] = matrix[0, col];
                for (int row = 0; row < rows; row++)
                {
                    if (matrix[row, col] < min[col])
                    {
                        min[col] = matrix[row, col];
                    }
                }
            }

            for (int row = 0; row < rows; row++)
            {
                for (int col = 0; col < cols; col++)
                {
                    Console.Write("{0, 8}", matrix[row, col]);
                }
                Console.WriteLine();
            }

            Console.WriteLine("Minimum:");
            for (int col = 0; col < cols; col++)
            {
                Console.Write("{0, 8}", min[col]);
            }
            Console.WriteLine();
        }
    }
}
```

Задача 6.4. Лотариен Билет

Прасчо си купил лотариен билет. Тъй като Прасчо не разбирал много-много, но пък имал голям късмет, отишъл при Мечо Пух да му помогне с „разшифроването“ на лотарийния билет. Лотарийния билет след

изтъркване представлява таблички от числа с n реда и m колони. Един билет печели, ако:

- Сумата от елементите намиращи се на главния диагонал е равна на сумата от елементите намиращи се на вторичния диагонал
- Сумата от елементите НАД главния диагонал е четна
- Сумата от елементите ПОД главния диагонал е нечетна
- Точната печалба се определя като средноаритметично от следните суми:
- Сума от всички елементи намиращи се ПОД главния диагонал
- Сумата на елементите, които са четни числа и се намират точно на главния диагонал
- Сумата на четните по стойност елементи, които са на външни редове (т.е. първи и последен)
- Сумата на нечетните по стойност елементи, които са на външни колони (т.е. първа и последна)

Нормално и напълно очаквано е едно и също число да принадлежи към повече от една от тези суми. Всички числа в таблицата са положителни цели числа.

От вас се очаква да изведете: YES, ако билетът печели, както и печалбата му, закръглена до втори знак след запетаята и NO, в противен случай.

Вход

Размерностите на таблицата ще бъдат въведени от един и същи ред, разделени с интервал.

Таблицата ще бъде въведена по редове, като всеки елемент на даден ред е разделен с интервал.

Примери

Вход	Изход	Обяснение
3 3 1 2 2 3 5 6 8 8 9	YES The amount of money won is: 13.00	Сумата от главния диагонал е 15, от вторичния също. Сумата на елементите над главния диагонал е 10 (2+2+6), сумата на елементите под главния диагонал е 19 (3+8+8), следователно билета изпълнява условията да е печелившь. Оттам нататък изчисляваме печалбата: • **Сумата под диагонала е 19 (3+8+8)** • **Сумата от четните елементи точно на главния диагонал е 0 (на**

		диагонала НЕ лежат четни елементи) • **На външните редове лежат следните четни числа:2+2+8+8=20** • **На външните колони лежат следните нечетни: 1+3+9=13** Тяхното средно аритметично е (19+0+20+13)/4 = 13.00

Подсказки

Въведете масива, след което го обходете. Правете следните проверки:

- Ако номерът на реда съвпада с номера на колоната, то елемента с тези индекси лежи НА главния диагонал
- Ако номерът на реда е по-малък от номера на колоната, то елемента с тези индекси лежи НАД главния диагонал
- Ако номерът на реда е по-голям от номера на колоната, то елемента с тези индекси лежи ПОД главния диагонал
- Ако сборът от номерата на реда и колоната е равен на броя на редовете – 1, то елемента с тези индекси лежи НА вторичния диагонал.
- Проверка за четен елемент може да извършите по подобен начин: lotteryTicket[row, col]%2==0
- **Проверка за четен елемент може да извършите по подобен начин: lotteryTicket[row, col]%2==1**

Решение

```
namespace LottaryTicket
{
    class Program
    {
        static void Main(string[] args)
        {
            var line = Console.ReadLine().Split().Select(int.Parse).ToArray();
            int n = line[0], m = line[1];
            int[,] ticket = new int[n, m];

            for (int row = 0; row < n; row++)
            {
                line = Console.ReadLine().Split().Select(int.Parse).ToArray();
                for (int col = 0; col < m; col++)
                {
                    ticket[row, col] = line[col];
```

```
    }
}

double d1 = 0, d2 = 0, d1upSum = 0, d1downSum = 0;
for (int x = 0; x < n; x++)
{
    for (int y = 0; y < m; y++)
    {
        if (x == y)
        {
            d1 += ticket[x, y];
        }
        else if (x > y)
        {
            d1downSum += ticket[x, y];
        }
        else if (y > x)
        {
            d1upSum += ticket[x, y];
        }
    }
}
for (int i = 0; i < m; i++)
{
    d2 += ticket[i, n - 1 - i];
}

if (d1 == d2 && d1upSum % 2 == 0 && d1downSum % 2 != 0)
{
    double d1even = 0;
    for (int x = 0; x < n; x++)
    {
        for (int y = 0; y < m; y++)
        {
            if (x == y && ticket[x, y] % 2 == 0)
            {
                d1even += ticket[x, y];
            }
        }
    }

    double extRowEven = 0;
    for (int j = 0; j < m; j++)
    {
        if (ticket[0, j] % 2 == 0)
        {
            extRowEven += ticket[0, j];
        }
        if (ticket[n - 1, j] % 2 == 0)
        {
            extRowEven += ticket[n - 1, j];
        }
    }

    double extColOdd = 0;
```

```
                for (int j = 0; j < n; j++)
                {
                    if (ticket[j, 0] % 2 != 0)
                    {
                        extColOdd += ticket[j, 0];
                    }
                    if (ticket[j, m - 1] % 2 != 0)
                    {
                        extColOdd += ticket[j, m - 1];
                    }
                }

                double profit = (d1downSum + d1even + extRowEven + extColOdd) / 4.0;
                Console.WriteLine("YES\nThe amount of money won is: {0:f2}",
profit);
            }
            else Console.WriteLine("NO");
        }
    }
}
```

Задача 6.5. Максимална площадка

Напишете програма, която въвежда квадратна матрица с цели числа. Намерете максималната под матрица с размери 2x2 и я отпечатайте в конзолата. Под максимална площадка се разбира такава под матрица с размер 2x2, така че сумата от нейните елементи да е максимална.

Примери

Вход	Изход
5 5 1 2 8 3 4 5 6 7 8 9 2 8 3 4 5 7 5 1 0 2 1 8 9 9 3	8 3 7 8

Упътване

- Обходете масива, пропускайки последния ред и последната колона. Ако в момента се намирате в елемент с индекси row, col, то площадката е с индекси:
  - row, col
  - row, col+1
  - row+1, col
  - row+1, col+1
- Сумирайте елементите на тези индекси и проверете дали сумата е по-голяма от досегашния резултат. Ако сте намерили нова по-голяма сума, запазете двата индекса на горния ляв ъгъл на площадката – row, col, за които се постига.
- Накрая изведете площадката, знаейки индекса на елемента, от която започва

Решение

```
namespace MaximumSite
{
    class Program
    {
        static void Main(string[] args)
        {
            var rows = int.Parse(Console.ReadLine());
            var cols = int.Parse(Console.ReadLine());

            int[,] matrix = new int[rows, cols];
            for (int row = 0; row < rows; row++)
            {
                var line = Console.ReadLine().Split().Select(int.Parse).ToArray();
                for (int col = 0; col < cols; col++)
                {
                    matrix[row, col] = line[col];
                }
            }

            int maxRow = 0, maxCol = 0;
            int maxSum = matrix[maxRow, maxCol];
            for (int row = 0; row < rows - 1; row++)
            {
                for (int col = 0; col < cols - 1; col++)
                {
                    var tempSum = matrix[row, col] +
                                  matrix[row, col + 1] +
                                  matrix[row + 1, col] +
                                  matrix[row + 1, col + 1];
                    if (tempSum > maxSum)
                    {
                        maxSum = tempSum;
                        maxRow = row;
                        maxCol = col;
                    }
                }
            }

            Console.WriteLine
            (
                "{0} {1}\n{2} {3}",
                matrix[maxRow, maxCol],
                matrix[maxRow, maxCol + 1],
                matrix[maxRow + 1, maxCol],
                matrix[maxRow + 1, maxCol + 1]
            );
        }
    }
}
```

Задача 6.6. Морски шах

Напишете програма, която въвежда конфигурация (3x3) на играта Морски Шах. По зададената конфигурация трябва да определите дали има

победител и ако има – да изведете кой е той. Символите в конфигурацията ще са следните: X, O, -, където – отбелязва позиция, на която НЕ Е поставен знак. Тестовите примери ще са такива, че да представляват завършена игра.

Примери

Вход	Изход	Вход	Изход	Вход	Изход
X - O - X O - - X	The winner is: X	X - O - X O - - O	The winner is: O	X O X X O O O X X	There is no winner

Решение

```
namespace TicTacToe
{
    class Program
    {
        static void Main(string[] args)
        {
            const int rows = 3, cols = 3;
            char[,] board = new char[rows, cols];

            for (int row = 0; row < rows; row++)
            {
                var line = Console.ReadLine().Split().Select(char.Parse).ToArray();
                for (int col = 0; col < cols; col++)
                {
                    board[row, col] = line[col];
                }
            }

            bool Xwin =
            (
                  (board[0, 0] == 'X' && board[1, 1] == 'X' && board[2, 2] == 'X')
||
                  (board[0, 2] == 'X' && board[1, 1] == 'X' && board[2, 0] == 'X')
||
                  (board[0, 0] == 'X' && board[0, 1] == 'X' && board[0, 2] == 'X')
||
                  (board[1, 0] == 'X' && board[1, 1] == 'X' && board[1, 2] == 'X')
||
                  (board[2, 0] == 'X' && board[2, 1] == 'X' && board[2, 2] == 'X')
||
                  (board[0, 0] == 'X' && board[1, 0] == 'X' && board[2, 0] == 'X')
||
                  (board[0, 1] == 'X' && board[1, 1] == 'X' && board[2, 1] == 'X')
||
                  (board[0, 2] == 'X' && board[1, 2] == 'X' && board[2, 2] == 'X')
                  ? true : false
            );
            bool Owin =
            (
```

```
                    (board[0, 0] == 'O' && board[1, 1] == 'O' && board[2, 2] == 'O')
||
                    (board[0, 2] == 'O' && board[1, 1] == 'O' && board[2, 0] == 'O')
||
                    (board[0, 0] == 'O' && board[0, 1] == 'O' && board[0, 2] == 'O')
||
                    (board[1, 0] == 'O' && board[1, 1] == 'O' && board[1, 2] == 'O')
||
                    (board[2, 0] == 'O' && board[2, 1] == 'O' && board[2, 2] == 'O')
||
                    (board[0, 0] == 'O' && board[1, 0] == 'O' && board[2, 0] == 'O')
||
                    (board[0, 1] == 'O' && board[1, 1] == 'O' && board[2, 1] == 'O')
||
                    (board[0, 2] == 'O' && board[1, 2] == 'O' && board[2, 2] == 'O')
                    ? true : false
            );

            if (Xwin)
            {
                Console.WriteLine("The winner is: X");
            }
            else if (Owin)
            {
                Console.WriteLine("The winner is: O");
            }
            else
            {
                Console.WriteLine("There is no winner");
            }
        }
    }
}
```

Задача 6.7. Триъгълник на Паскал

Генерирайте и отпечатайте Триъгълника на Паскал по зададена височина h. Триъгълника на Паскал съдържа:

- Числото 1 на 1 ред
- Всяко число на всеки следващ ред се получава от сбора на двете числа над него

Примери

Вход	Изход	Вход	Изход
5	1 1 1 1 2 1 1 3 3 1 1 4 6 4 1	4	1 1 1 1 2 1

Решение

```
namespace PascalTriangle
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            int h = int.Parse(Console.ReadLine());

            long[][] triangle = new long[h + 1][];
            for (int row = 0; row < h; row++)
            {
                triangle[row] = new long[row + 1];
            }

            triangle[0][0] = 1;

            for (int row = 0; row < h - 1; row++)
            {
                for (int col = 0; col <= row; col++)
                {
                    triangle[row + 1][col] += triangle[row][col];
                    triangle[row + 1][col + 1] += triangle[row][col];
                }
            }

            for (int row = 0; row < h; row++)
            {
                Console.Write(new String(' ', 2 * (h - row)));
                for (int col = 0; col <= row; col++)
                {
                    Console.Write("{0,3} ", triangle[row][col]);
                }
                Console.WriteLine();
            }
        }
    }
}
```

Задача 6.8. Таблички

Иванчо много обича да си прави таблички в Excel. Понеже Иванчо е Excel Master си прави и Sheet-ове. На Иванчо обаче това не му стига. Затова той ще ви даде данните от един файл с всичките му Sheet-ове, от вас се иска да изведете малко статистика:

- От всеки sheet:
  - Минимален елемент
  - Максимален елемент
  - Средноаритметично
- Обобщено:
  - Средноаритметично на целия документ – получава се като разделим средноаритметичните от всеки sheet на броя на sheet-овете
  - Колко елемента от всеки sheet са над средноаритметичното за целия документ.

Вход

Данните ще са въведени по следния начин:

- На първи ред въвеждате цяло число n - броя на sheet-овете
- Следват n групи редове – всяка група се състои от:
  - Един ред с две стойности разделени с интервал – броя на редовете и колоните в текущия sheet
  - Таблицата от sheet-а – състои се от цели числа разделени с интервали

Изход

Резултата от програмата се извежда по следния начин:

- Първо извеждате n реда – на всеки от тях по 3 числа, разделени в интервал в следния ред: минимум, максимум, средноаритметично на съответния sheet
- След това извеждате един ред с n на брой елемента, разделени с интервал – съответния брой числа над средноаритметичното за всеки един sheet
- Всички реални числа се закръглят до 2 знак след запетаята

Примери

Вход	Изход
3 3 3 1 2 3 4 5 6 7 8 9 2 2 4 5 8 9 3 3 1 2 8 8 5 4 3 2 8	1 9 5 4 9 6.5 1 8 4.56 4 2 3

Подсказки

- Създайте си назъбен масив, чиито елементи са двумерни масиви: int[][,] document = new int[sheets][,]

Решение

```
namespace Sheets
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            int[][,] document = new int[n][,];
            for (int i = 0; i < n; i++)
            {
                var line = Console.ReadLine().Split().Select(int.Parse).ToArray();
                document[i] = new int[line[0], line[1]];
```

```
                for (int j = 0; j < line[0]; j++)
                {
                    var input =
Console.ReadLine().Split().Select(int.Parse).ToArray();
                    for (int q = 0; q < line[1]; q++)
                    {
                        document[i][j, q] = input[q];
                    }
                }
            }
            double globalAvg = 0, localAvg = 0;
            int max = 0, min = 0;
            for (int i = 0; i < n; i++)
            {
                localAvg = 0;
                max = min = document[i][0, 0];
                for (int j = 0; j < document[i].GetLength(0); j++)
                {
                    for (int q = 0; q < document[i].GetLength(1); q++)
                    {
                        localAvg += document[i][j, q];
                        if (max < document[i][j, q]) max = document[i][j, q];
                        if (min > document[i][j, q]) min = document[i][j, q];
                    }
                }
                localAvg = localAvg / (document[i].GetLength(0) *
document[i].GetLength(1));
                Console.WriteLine("{0} {1} {2}", min, max, Math.Round(localAvg, 2));
                globalAvg += localAvg;
            }
            globalAvg /= n;
            for (int i = 0; i < n; i++)
            {
                int count = 0;
                for (int j = 0; j < document[i].GetLength(0); j++)
                {
                    for (int q = 0; q < document[i].GetLength(1); q++)
                    {
                        if (document[i][j, q] > globalAvg) count++;
                    }
                }
                Console.Write("{0} ", count);
            }
            Console.WriteLine();
        }
    }
}
```

Тема 7. Речници и хеш таблици

Задача 7.1. Нечетни срещания

Напишете програма, която извлича от поредица от думи всички елементи, които се срещат нечетен брой пъти (без значение от големината на буквите)

- Думите са въведени на един ред разделени с интервал
- Изведете получените думи с малки букви, по реда им на поява

Примери

Вход	Изход
Java C# PHP PHP JAVA C java	java, c#, c
3 5 5 hi pi HO Hi 5 ho 3 hi pi	5, hi
a a A SQL xx a xx a A a XX c	a, sql, xx, c

Подсказки

- Използвайте речник (string → int), за да преброите всички срещания на всяка дума (като в предната задача).
- Обходете всички двойки ключ-стойност в речника и запазете в резултатите всички ключове, които имат нечетна стойност.
- Изведете списъка с резултатите

Решение

```
namespace OddOccurrences
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] words = Console.ReadLine().ToLower().Split(' ');
            var counts = new Dictionary<string, int>();

            foreach (var word in words)
            {
                if (counts.ContainsKey(word))
                {
                    counts[word]++;
                }
                else
                {
                    counts[word] = 1;
                }
            }

            var results = new List<string>();
            foreach (var pair in counts)
            {
                if (pair.Value % 2 != 0)
                {
                    results.Add(pair.Key);
                }
            }

            Console.WriteLine(string.Join(", ", results));
        }
    }
}
```

Задача 7.2. Телефонен указател

Напишете програма, която получава информация от конзолата за хора и техните телефонни номера. Всеки запис трябва да има само едно име и телефон (и двете се пазят в низ).

На всеки ред ще получите някоя от следните команди:

- A {име} {телефон} – добавя записа към телефонния указател. В случай че се добавя име, което вече съществува в указателя трябва да смените съществуващия номер с новия.
- S {име} – търси се човек с такива име и се извежда резултат във формат "{име} -> {номер}". В случай на несъществуващ контакт, изведете "Contact {име} does not exist.".
- END – спира получаването на команди.

Пример

Вход	Изход
A Nakov 0888080808 S Mariika S Nakov END	Contact Mariika does not exist. Nakov -> 0888080808
A Nakov +359888001122 A RoYaL(Ivan) 666 A Gero 5559393 A Simo 02/987665544 S Simo S simo S RoYaL S RoYaL(Ivan) END	Simo -> 02/987665544 Contact simo does not exist. Contact RoYaL does not exist. RoYaL(Ivan) -> 666
A Misho +359883123 A Misho 02/3123 S Misho END	Misho -> 02/3123

Подсказки

- Обработвайте командите като ги разделяте чрез интервал. Изпълнявайте командите докато не стигнете до "END".
- Съхранявайте указателя в Dictionary<string, string> с ключ {име} и стойност {телефонен номер}.

Решение

```
namespace PhoneBook
{
    class Program
    {
```

```
        static void Main(string[] args)
        {
            var book = new Dictionary<string, string>();
            while (true)
            {
                var line = Console.ReadLine().Split(' ');
                switch (line[0])
                {
                    case "A":
                        {
                            book[line[1]] = line[2];
                            break;
                        }
                    case "S":
                        {
                            if (book.ContainsKey(line[1]))
                            {
                                Console.WriteLine("{0} -> {1}", line[1],
book[line[1]]);
                            }
                            else
                            {
                                Console.WriteLine("Contact {0} does not exist.",
line[1]);
                            }
                            break;
                        }
                    case "END": return;
                }
            }
        }
    }
}
```

Задача 7.3. Миньорска задача

Получавате поредица от низове, всеки на нов ред. Всеки нечетен ред на конзолата показва полезно изкопаемо (злато, сребро, мед и т.н.), а всеки четен - количество. Вашата задача е да съберете изкопаемите и да изпечатате всяко на нов ред.

Изведете изкопаемите и техните количества във формат:

{resource} –> {quantity}

Количествата ще бъдат в интервала [1 ... 2 000 000 000]

Примери

Input	Output

Input	Output

Gold 155 Silver 10 Copper 17 stop	Gold -> 155 Silver -> 10 Copper -> 17

gold 155 silver 10 copper 17 gold 15 stop	gold -> 170 silver -> 10 copper -> 17

Решение

```
namespace MiningTask
{
    class Program
    {
        static void Main(string[] args)
        {
            var goldMiners = new Dictionary<string, int>();

            while (true)
            {
                string key = Console.ReadLine();
                if (key == "stop") break;
                int value = int.Parse(Console.ReadLine());
                goldMiners.Add(key, value);
            }

            foreach (var pair in goldMiners)
            {
                Console.WriteLine("{0} -> {1}", pair.Key, pair.Value);
            }
        }
    }
}
```

Задача 7.4. Супермаркет

Напишете програма, която пази информация за продукти и техните цени. Всеки продукт си има име, цена и количество. Ако продуктът не съществува в базата данни, той се добавя със стартово количество.

Ако получите продукт, който вече съществува, то ще увеличите неговото количество и ако цената е различна, ще замените старата с новата цена.

Ще получите име, цена и количество за всеки продукт на нов ред. Накрая ще стои команда "stocked". При нейното срещане, изведете всичките артикули с техните име, цена, наличност и обща цена на вски продукти с това име. Когато изведете всички продукти, изведете и общата цена на всички артикули.

Важно: Общата цена се изчислява на базата на най-новата цена за всеки продукт

Вход

- Докато не получите "stocked", ще получавате продукту във формат: "{име} {цена} {количество}".
- Данните ще бъдат винаги отделени от един интервал

Изход

- Изведете информацията за всеки продукт, следвайки формата: "{име}: ${цена:F2} * {количество} = ${общо:F2}"
- На следващия ред, изведете 30 тиренца.
- На последния ред, изведете общо: "Grand Total: ${grandTotal:F2}"

Примери

Вход	Изход
Beer 2.20 100 IceTea 1.50 50 NukaCola 3.30 80 Water 1.00 500 stocked	Beer: $2.20 * 100 = $220.00 IceTea: $1.50 * 50 = $75.00 NukaCola: $3.30 * 80 = $264.00 Water: $1.00 * 500 = $500.00 ------------------------------ Grand Total: $1059.00
Beer 2.40 350 Water 1.25 200 IceTea 5.20 100 Beer 1.20 200 IceTea 0.50 120 stocked	Beer: $1.20 * 550 = $660.00 Water: $1.25 * 200 = $250.00 IceTea: $0.50 * 220 = $110.00 ------------------------------ Grand Total: $1020.00
CesarSalad 10.20 25 SuperEnergy 0.80 400 EvenSupererEnergy 1.00 400 Beer 1.35 350 beer 0.50 450 IceCream 1.50 25 stocked	CesarSalad: $10.20 * 25 = $255.00 SuperEnergy: $0.80 * 400 = $320.00 EvenSupererEnergy: $1.00 * 400 = $400.00 Beer: $1.35 * 350 = $472.50 beer: $0.50 * 450 = $225.00 IceCream: $1.50 * 25 = $37.50 ------------------------------ Grand Total: $1710.00

Решение

```
namespace SuperMarket
{
    class Program
    {
        static void Main(string[] args)
        {
            var Market = new Dictionary<string, Tuple<float, int>>();

            while (true)
```

```
            {
                var line = Console.ReadLine().Split(' ');
                if (line[0] == "stocked") break;

                float key = float.Parse(line[1]);
                int value = int.Parse(line[2]);
                Market.Add(line[0], new Tuple<float, int>(key, value));
            }

            double total = 0;
            foreach (var pair in Market)
            {
                float sum = pair.Value.Item1 * pair.Value.Item2;
                Console.WriteLine("{0}: ${1:f2} * {2} = ${3:f2}",
                                  pair.Key, pair.Value.Item1, pair.Value.Item2,
sum);
                total += sum;
            }

            Console.WriteLine(new String('-', 30));
            Console.WriteLine("Grand Total: ${0:f2}", total);
        }
    }
}
```

Задача 7.5. Брой на реалните числа

Въведете списък от реални числа и ги изведете в нарастващ ред заедно с броя на срещанията им.

Примери

Вход	Изход
8 2.5 2.5 8 2.5	2.5 -> 3 8 -> 2

Вход	Изход
1.5 5 1.5 3	1.5 -> 2 3 -> 1 5 -> 1

Вход	Изход
-2 0.33 0.33 2	-2 -> 1 0.33 -> 2 2 -> 1

Подсказки

- Използвайте **SortedDictionary<double, int>** с име **counts**.
- Обходете всяко число **num** и увеличете **counts[num]** (когато **num** съществува в речника) или присвоете **counts[num]** = **1** (когато **num** не съществува в речника).
- Обходете всяко едно от числата **num** в речника (**counts.Keys**) и изведете числото **num** и броя на срещанията му **counts[num]**.

Решение

```
namespace RealNumbersCount
{
    class Program
    {
        static void Main(string[] args)
        {
            var nums = Console.ReadLine().Split().Select(float.Parse);
```

```
            var counts = new SortedDictionary<float, int>();

            foreach (var num in nums)
            {
                if (counts.ContainsKey(num))
                {
                    counts[num]++;
                }
                else
                {
                    counts[num] = 1;
                }
            }

            foreach (var pair in counts)
            {
                Console.WriteLine("{0} -> {1}", pair.Key, pair.Value);
            }
        }
    }
}
```

Задача 7.6. Подобрен телефонен указател

Добавете функционалност към указателя от предната задача да извежда всички контакти в азбучен ред, когато получи командата ListAll.

Примери

Вход	Изход
A Nakov +359888001122 A RoYaL(Ivan) 666 A Gero 5559393 A Simo 02/987665544 ListAll END	Gero -> 5559393 Nakov -> +359888001122 RoYaL(Ivan) -> 666 Simo -> 02/987665544

Подсказки

1. Начин I (по-бавен): Сортирайте всички записи в речника по ключ и ги изведете.
2. Начин II (по-бърз): Пазете записите в по-подходяща структура, която да ги пази в сортиран ред по начало.

Решение

```
namespace AdvancedPhoneBook
{
    class Program
    {
        static void Main(string[] args)
        {
            var book = new SortedDictionary<string, string>();
            while (true)
            {
                var line = Console.ReadLine().Split(' ');
                switch (line[0])
```

```
                {
                    case "A":
                        {
                            book[line[1]] = line[2];
                            break;
                        }
                    case "S":
                        {
                            if (book.ContainsKey(line[1]))
                            {
                                Console.WriteLine("{0} -> {1}", line[1],
book[line[1]]);
                            }
                            else
                            {
                                Console.WriteLine("Contact {0} does not exist.",
line[1]);
                            }
                            break;
                        }
                    case "ListAll":
                        {
                            foreach (var pair in book)
                            {
                                Console.WriteLine("{0} -> {1}", pair.Key,
pair.Value);
                            }
                            break;
                        }
                    case "END": return;
                }
            }
        }
    }
}
```

Задача 7.7. Сума, минимум, максимум, средноаритметично

Напишете програма, която въвежда n цели числа и извежда тяхната сума, минимум, максимум, първи елемент, последен елемент и средноаритметично.

Примери

Вход	Изход
5 12 20 -5 37 8	Sum = 72 Min = -5 Max = 37 Average = 14.4

4 50 20 25 40	Sum = 135 Min = 20 Max = 50 Average = 33.75

Подсказки

- Добавете "**System.Linq**", за да може да ползвате функциите от LINQ.
- Въведете масива от входни числа **nums[]**.
- Използвайте **nums.Min()**, **nums.Max()**, и т.н.

Решение

```
namespace SumMinMaxAvr
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            int[] nums = new int[n];
            for (int i = 0; i < n; i++) nums[i] = int.Parse(Console.ReadLine());

            Console.WriteLine("Sum = {0}", nums.Sum());
            Console.WriteLine("Min = {0}", nums.Min());
            Console.WriteLine("Max = {0}", nums.Max());
            Console.WriteLine("Average = {0}", nums.Average());
        }
    }
}
```

Задача 7.8. Трите най-големи числа

Въведете списък от реални числа и изведете 3 най-големи от тях. Ако по-малко от 3 числа съществуват, изведете всички от тях.

Примери

Вход	Изход
10 30 15 20 50 5	50 30 20

Вход	Изход
20 30	30 20

Подсказки

Може да използвате LINQ заявка по следния начин:
nums.OrderByDescending(x => x).Take(3).

Решение

```
namespace ThreeLargestNumbers
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
            double[] nums = Console.ReadLine().Split('
').Select(double.Parse).ToArray();
            nums = nums.OrderByDescending(num => num).Take(3).ToArray();
            Console.Write(string.Join(" ", nums));
        }
    }
}
```

Задача 7.9. Сортиране на кратки думи

Въведете текст, извлечете неговите думи, намерете всички кратки думи (с по-малко от 5 знака) и ги отпечатайте в азбучен ред, с малки букви.

- Използвайте следните разделители: . , : ; ( ) [ ] " ' \ / ! ? *(space)*.
- Големината на буквите няма значение
- Премахнете дублиращите се думи.

Примери

Вход	Изход
In CodeCamp you can study Java, C#, PHP and JavaScript. JAVA and c# developers graduate in 2-3 years. Go in!	2-3, and, c#, can, go, in, java, php, you

Подсказки

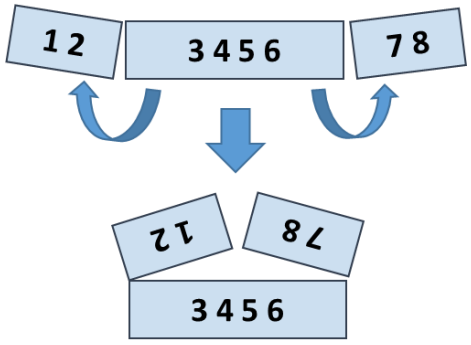
- За да извлечете думите от входния текст, разделете го чрез зададените разделители.
- Използвайте LINQ изрази:
  - Филтрирайте по дължина на думата: **`Where(…)`**
  - Подредете думите: **`OrderBy(…)`**
  - Използвайте **`Distinct()`**, за да премахнете дублиранията

Решение

```
namespace ShortWordsSort
{
    class Program
    {
        static void Main(string[] args)
        {
            char[] separators = new char[]
            {'.',',',':',';','(',')','[',']','\\','\"','\'','/','!','?',' '};
            string sentence = Console.ReadLine().ToLower();
            string[] words = sentence.Split(separators);
            var result = words.Where(w => w != "").OrderBy(w => w).Distinct();
            result = result.Where(x => x.Length > 5).ToArray();
            Console.WriteLine(string.Join(", ", result));
        }
    }
}
```

Задача 7.10. Сгъни и сумирай

Въведете масив от 4*k цели числа, сгънете го както е показано по-долу и изведете сумата на горния и долния ред (2*k цели числа):



Примери

Вход	Изход	Коментари
5 2 3 6	7 9	5 6 + 2 3 = 7 9
1 2 3 4 5 6 7 8	5 5 13 13	2 1 8 7 + 3 4 5 6 = 5 5 13 13
4 3 -1 2 5 0 1 9 8 6 7 -2	1 8 4 -1 16 14	-1 3 4 -2 7 6 + 2 5 0 1 9 8 = 1 8 4 -1 16 14

Подсказки

Използвайте LINQ израз:

- Ред 1, лява част: вземете първите **k** числа и обърнете наобратно.
- Ред 1, дясна част: обърнете наобратно и вземете първите **k** числа.
- Слейте лявата и дясната част на ред 1.
- Ред 2: пропуснете първите k числа и вземете следващите 2*k числа.
- Сумирайте масивите **row1** и **row2**: **var sum = row1.Select((x, index) => x + row2[index])**.

Решение

```
namespace FoldAndSum
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] input = Console.ReadLine().Split(' ').Select(int.Parse).ToArray();
            int k = input.Length / 4;
```

```
            int[] row1left = input.Take(k).Reverse().ToArray();
            int[] row1right = input.Reverse().Take(k).ToArray();
            int[] row1 = row1left.Concat(row1right).ToArray();
            int[] row2 = input.Skip(k).Take(2 * k).ToArray();
            var sum = row1.Select((x, index) => x + row2[index]);
            Console.WriteLine(String.Join(" ", sum));
        }
    }
}
```

Задача 7.11. Сортиране на часове

Напишете програма, която получава списък от часове (разделени с интервал, 24-часове формат) и ги сортира в нарастващ ред. Изведете сортираните часове разделени с интервали.

Примери: 06:55, 02:30, 23:11 ➔ 02:30, 06:55, 21:11

Пример

Вход	Изход
00:00 06:04 02:59 10:33 11:22 06:01	00:00, 02:59, 06:01, 06:04, 10:33, 11:22
04:25 04:21 04:19	04:19, 04:21, 04:25
00:00 23:59 12:00 16:00	00:00, 12:00, 16:00, 23:59

Решение

```
namespace TimeSort
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] item = Console.ReadLine().Split(' ');
            item = item.OrderBy(x => x).Distinct().ToArray();
            Console.WriteLine(string.Join(" ", item));
        }
    }
}
```

Задача 7.12. Нечетен филтър

Напишете програма, която получава масив от цели числа (раздели с интервал), премахва всички нечетни числа, след което превръща останалите числа в нечетни числа, според:

- Ако числото е по-голямо от средното от колекцията оставащи числа, добавяме 1
- Ако числото е по-малко от средното от колекцията оставащи числа, то намаляме с 1.

След като превърнете всички елементи към нечетни числа, изведете масива

Примери

Вход	Изход
1 2 3 4 5 6 7 8 9 10	1 3 5 9 11
99 88 77 66 55 4 33 22 11	89 67 3 21
22 **2 199 723 8127 95**	31

Решение

```
namespace OddFilter
{
    class Program
    {
        static void Main(string[] args)
        {
            var num = Console.ReadLine().Split(' ').Select(int.Parse).ToArray();

            var even = num.Where(x => x % 2 == 0).ToArray();
            var avg = even.Average();

            var num1 = even.Where(x => x <= avg).Select(y => y = y - 1).ToArray();
            var num2 = even.Where(x => x > avg).Select(y => y = y + 1).ToArray();
            var num3 = num1.Concat(num2);

            Console.WriteLine(string.Join(" ", num3));
        }
    }
}
```

Задача 7.13. Имунна система

Всеки организъм може да бъде нападнат от различни видове вируси. Информация за тях се съхраняват в имунната му система. Ако тя вече е среща вируса, ще го победи по-бързо, отколкото ако го среща за първи път.

Имунната система може да изчисли силата на вируса преди да го победи. Това е сумата от кодовете на всички букви в името на вируса, разделена на 3.

Имунната система може да изчисли времето, което трябва да се победи даден вирус в секунди. То е равно на силата на вируса умножено по дължината на името на вируса.

Когато изчислите времето за побеждаване на вируса, превърнете го в минути и секунди, като не използвате водещи нули.

- Ако имунната система победи вируса, изведете:
  "{virusName} defeated in {virusDefeatMinutes}m {virusDefeatSeconds}s."
- Ако силата на вируса е по-голяма от силата на имунната система, изведете "**`Immune System Defeated.`**" И приключете изпълнението на програмата.

След като вируса е победен, имунната система си възвръща 20% от нейната сила. Ако 20-те процента надвишават първоначалната сила на имунната система, задайте състоянието ѝ до нейната първоначална сила.

Пример: вирус "**flu1**":

- сила: 102 (f) + 108 (l) + 117 (u) + 49 (1) = 376 / 3 = 125.33 = 125.
- Време за победа: 125 * 4 (дължина на името) = 500 секунди ➔ 8m 20s.

Пример 2: Срещна на "flu1" за втори път:

- Време за победа: (125 * 4) / 3 = 166.66 ➔ 166 секунди

Ако се срещне един и същ вирус последователно, НЕ намаляйте времето за неговата победа допълнително. Когато получите "end", изведете статуса на имунната система във формата "Final Health: {finalHealth}".

Вход

- На първи ред: началната сила на имунната система
- На всеки нов ред, до срещане на "end": имена на вируси

Изход

Изход при победен вирус изглежда така:

- Първи ред: "`Virus {virusName}: {virusStrength} => {virusDefeatSeconds}`"
- Втори ред: "`{virusName} defeated in {defeatMins}m {defeatSecs}s.`"
- Трети ред: "`Remaining health: {remainingHealth}`". Оставащото здраве се извежда преди да се регенерира.

Примери

Вход	Изход
5000 flu1 test flu1 virusssssss end	Virus flu1: 125 => 500 seconds flu1 defeated in 8m 20s. Remaining health: 4500 Virus test: 149 => 596 seconds test defeated in 9m 56s. Remaining health: 4404 Virus flu1: 125 => 166 seconds flu1 defeated in 2m 46s. Remaining health: 4834 Virus virusssssss: 419 => 4609 seconds virusssssss defeated in 76m 49s. Remaining health: 391 Final Health: 469
1750	Virus Ebola: 161 => 805 seconds

Ebola ebola Ebola end	Ebola defeated in 13m 25s. Remaining health: 945 Virus ebola: 171 => 855 seconds ebola defeated in 14m 15s. Remaining health: 279 Virus Ebola: 161 => 268 seconds Ebola defeated in 4m 28s. Remaining health: 66 Final Health: 79
5700 wannacry iskaplache wannacry	Virus wannacry: 289 => 2312 seconds wannacry defeated in 38m 32s. Remaining health: 3388 Virus iskaplache: 348 => 3480 seconds iskaplache defeated in 58m 0s. Remaining health: 585 Virus wannacry: 289 => 770 seconds Immune System Defeated.

Решение

```
namespace ImmuneSystem
{
    class Program
    {
        static void Main(string[] args)
        {
            int health = int.Parse(Console.ReadLine());
            int originalhealth = health;
            string virus = Console.ReadLine();
            var encountered = new Dictionary<string, int>();
            string previousvirus = string.Empty;

            while (true)
            {
                if (encountered.ContainsKey(virus))
                {
                    encountered[virus] += 1;
                }
                else
                {
                    encountered[virus] = 1;
                }

                int power = 0;
                foreach (var ch in virus)
                {
                    power += (int)ch;
                }
                power /= 3;

                int time = power * virus.Length;
```

```
                if (encountered[virus] > 1 && previousvirus != virus)
                {
                    time /= 3;
                }
                health -= time;

                Console.WriteLine($"Virus {virus}: {power} => {time} seconds");

                if (health <= 0)
                {
                    Console.WriteLine("Immune System Defeated.");
                    break;
                }
                else
                {
                    Console.WriteLine($"{virus} defeated in {time / 60}m {time %
60}s.");
                    Console.WriteLine($"Remaining health: {health}");
                    health = health + (health * 20) / 100;
                    if (health > originalhealth)
                    {
                        health = originalhealth;
                    }
                }

                previousvirus = virus;
                virus = Console.ReadLine();

                if (virus == "end")
                {
                    Console.WriteLine($"Final Health: {health}");
                    break;
                }
            }
        }
    }
}
```

Задача 7.14. Поправка на Email

Напишете програма, в която получавате последователност от низове, всеки на нов ред, докато срещнете команда "stop". Първия низ е името на човека. На втори ред, ще получите имейл. Вашата задача е да съберете техните имена и имейли, след което трябва да премахнете имейлите, чиито домейни завършват с "us" или "uk" (без значение от големината на буквите). Извеждайте в следния формат:

{name} – > {email}

Примери

Вход	Изход

Ivan ivanivan@abv.bg Petar Ivanov petartudjarov@abv.bg Mike Tyson myke@gmail.us stop	Ivan -> ivanivan@abv.bg Petar Ivanov -> petartudjarov@abv.bg

Решение

```
namespace EmailValidator
{
    class Program
    {
        static void Main(string[] args)
        {
            // v1
            while (true)
            {
                string[] commands = Console.ReadLine().Split(' ').ToArray();
                if (commands[0] == "stop")
                {
                    break;
                }
                string emails = Console.ReadLine();
                string names = (string.Join(" ", commands));
                if (emails.Contains("bg") && commands[0] != "stop")
                {
                    Console.WriteLine("{0} -> {1}", names, emails);
                }
            }

            // v2
            var emails = new Dictionary<string, string>();
            int row = 1;
            string line = Console.ReadLine();
            string name = line;
            while (line != "stop")
            {
                if (row % 2 == 0)
                {
                    emails[name] = line;
                }
                else
                {
                    name = line;
                }
                line = Console.ReadLine();
                row++;
            }
            foreach (var pair in emails)
            {
```

```
                if (!(pair.Value.ToLower().Contains("us") ||
pair.Value.ToLower().Contains("uk")))
                {
                    Console.WriteLine($"{pair.Key} -> {pair.Value}");
                }
            }
        }
    }
}
```

Задача 7.15. Добра ръка

Напишете програма, която въвежда поредица от хора, като за всеки човек се въвежда какви карти е изтеглил от тестето. Входните данни ще бъдат на отделни редове в следния формат:

- {име}: {PT, PT, PT,... PT}

Където P (2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A) е номера/буквата на картата, а T (S, H, D, C) е вида. Входа приключва тогава, когато се изтегли "JOKER". Името може да съдържа всякакви ASCII символи, освен ':'. Входът винаги ще бъде валиден, според описания формат и няма нужда да го проверявате.

Един човек не може да има повече от една карта от един и същ номер и вид, ако изтеглят такава карта, те я отказват. Хората играят с няколко тестета. Всяка карта има стойност, която е изчислена от номера умножен по типа. Номерата от 2 до 10 си имат същите стойности, а буквите от J до A са 11 до 14. Типовете съответстват на числа по следния начин: S -> 4, H-> 3, D -> 2, C -> 1.

Накрая изведете общата стойност, която всеки играч има в ръката си във формата:

- {име}: {стойност}

Пример

Вход	Изход
Pesho: 2C, 4H, 9H, AS, QS Slav: 3H, 10S, JC, KD, 5S, 10S Peshoslav: QH, QC, QS, QD Slav: 6H, 7S, KC, KD, 5S, 10C Peshoslav: QH, QC, JS, JD, JC Pesho: JD, JD, JD, JD, JD, JD JOKER	Pesho: 167 Slav: 175 Peshoslav: 197

Решение

Задача 7.16. Потребителски логове

Мариян е известен сисадмин. Все още не се е родил човека, който може да пробие неговите сървъри. Все пак, има нов вид заплаха, при която

потребители наводняват сървъра със съобщение и е трудно да бъдат засечени, понеже си менят IP адресите. Понеже Мариян е сисадмин, а не програмист, той ще има нужда от опитен програмист, за да проследи логовете. Сега топката е у вас!

Ще получите вход в следния формат:

- **IP=(IP.Address) message=(A&sample&message) user=(username)**

Вашата задача е да извлечете IP-то и потребителското име от входните данни за всеки потребител, трябва да покажете всяко IP, от което съответния потребител е изпратил съобщение и да преброите съобщенията изпратени със съответното IP. В изхода, потребителските имена трябва да бъдат сортирани азбучно, докато техните IP адреси трябва да бъдат сортирани по реда на тяхното първо появяване. Изходът трябва да бъде в следния формат:

```
Потребителско име:
IP => брой, IP => брой.
```

Например, даден е следния вход:

- **"IP=192.23.30.40 message='Hello&derps.' user=destroyer",**

Ще извлечете потребителското име destroyer и IP 192.23.30.40 и ще го покажете в следния формат:

```
destroyer:
192.23.30.40 => 1.
```

Потребител destroyer е изпратил съобщение от IP 192.23.30.40 веднъж.

Разгледайте примерите по-долу. Те ще изяснят условието.

Вход

Входът ще бъде подаден като вариращ брой редове. Ще трябва да обработвате всяка команда, докато получите команда end. Входът ще бъде във формата показан по-горе, няма нужда да го проверявате.

Изход

За всеки намерен потребител, ще трябва да изведете всеки лог във формата:

```
потребител:
IP => брой, IP => брой…
```

IP адресите трябва да бъдат разделени със запетая и всеки IP адрес трябва да приключва с точка.

Ограничения

- Броят на командите ще бъде в интервала [1..50]
- IP адресите ще бъдат във формата на IPv4 или IPv6.
- Съобщението ще е във формата: This&is&a&message
- Потребителското име ще бъде низ с дължина [3..50]
- Времеви лимит: 0.3 секунди. Лимит по памет: 16 MB.

Примери

Вход	Изход
IP=192.23.30.40 message='Hello&derps.' user=destroyer IP=192.23.30.41 message='Hello&yall.' user=destroyer IP=192.23.30.40 message='Hello&hi.' user=destroyer IP=192.23.30.42 message='Hello&Dudes.' user=destroyer end	destroyer: 192.23.30.40 => 2, 192.23.30.41 => 1, 192.23.30.42 => 1.
IP=FE80:0000:0000:0000:0202:B3FF:FE1E:8329 message='Hey&son' user=mother IP=192.23.33.40 message='Hi&mom!' user=child0 IP=192.23.30.40 message='Hi&from&me&too' user=child1 IP=192.23.30.42 message='spam' user=destroyer IP=192.23.30.42 message='spam' user=destroyer IP=192.23.50.40 message='' user=yetAnotherUsername IP=192.23.50.40 message='comment' user=yetAnotherUsername IP=192.23.155.40 message='Hello.' user=unknown end	child0: 192.23.33.40 => 1. child1: 192.23.30.40 => 1. destroyer: 192.23.30.42 => 2. mother: FE80:0000:0000:0000:0202:B 3FF:FE1E:8329 => 1. unknown: 192.23.155.40 => 1. yetAnotherUsername: 192.23.50.40 **> 2.**

Решение

Задача 7.17. Преброяване на населението

Толкова много хора! Толкова е трудно да ги преброиш. Но това е работата ви на статистик. Ще получите сурови данни за даден град, а вие трябва да ги обработите.

На всеки ред, ще получите информация във формата: "град|държава|население". Няма да има излишни интервали никъде във входните данни. Обработете данните по държава и град и ги изведете в конзолата.

За всяка държава, изведете нейното общо население и на отделни редове данните за всеки град. Държавите се подреждат по тяхното общо население в намалящ ред, а в рамките на всяка държава градовете се подреждат по същия начин.

Ако две държави/град имат едно и също население, пазете ги по реда им на въвеждане. Разгледайте примерите и следвайте точно форматирането!

Вход

- Входните данни се въвеждат от конзолата
- Състоят се от променлив брой редове и приключва, когато се извика команда "report".
- Входните данни винаги ще бъдат валидни и ще следват описания формат. Няма нужда да бъдат проверявани.

Изход

- Изходът трябва да бъде отпечатан на конзолата.
- Изведете обработената информация за всяка държава и град, според форматирането от примерите.

Ограничения

- Името на града, страната и броя на населението ще бъдат отделени с права черта ('|').
- Броят на редовете ще е в интервала [2 ... 50].
- Двойка град-държава няма да бъде повторена.
- Броят на населението във всеки град ще бъде цяло число в интервала [0 ... 2 000 000 000].
- Лимит по време: 0.1 секунди. Лимит по памет: 16 MB.

Примери

Вход	Изход
Sofia|Bulgaria|1000000 report	Bulgaria (total population: 1000000) =>Sofia: 1000000

Вход	Изход
Sofia|Bulgaria|1 Veliko Tarnovo|Bulgaria|2 London|UK|4 Rome|Italy|3 report	UK (total population: 4) =>London: 4 Bulgaria (total population: 3) =>Veliko Tarnovo: 2 =>Sofia: 1 Italy (total population: 3) =>Rome: 3

Решение

```
namespace PopulationCount
{
    public class Program
    {
```

```
        static void Main(string[] args)
        {
            Dictionary<string, Dictionary<string, int>> items = new
Dictionary<string, Dictionary<string, int>>();

            while (true)
            {
                var line = Console.ReadLine().Split("|");
                if (line[0] == "report") break;

                var city = line[0];
                var country = line[1];
                var population = int.Parse(line[2]);

                if (items.ContainsKey(country))
                {
                    var towns = items[country];
                    towns.Add(city, population);
                }
                else
                {
                    var town = new Dictionary<string, int>();
                    town.Add(city, population);
                    items.Add(country, town);
                }
            }

            foreach (var item in items)
            {
                Console.WriteLine($"{item.Key} (total population: {item.Value.Sum(x
=> x.Value)})");
                foreach (var town in item.Value)
                {
                    Console.WriteLine($"=>{town.Key}: {town.Value}");
                }
            }

        }
    }
}
```

Задача 7.18. Обработка на логове

Ще получите логове в следния формат <IP> <потребител> <продължителност>. Например:

- 192.168.0.11 peter 33
- 10.10.17.33 alex 12
- 10.10.17.35 peter 30
- 10.10.17.34 peter 120
- 10.10.17.34 peter 120
- 212.50.118.81 alex 46
- 212.50.118.81 alex 4

Напишете програма, която обработва логовете и извежда за всеки потребител общата продължителност на неговите сесии и списък от уникални IP адреси във формата: "<потребител>: <продължителност> [<IP1>, <IP2>, ...]". Подредете потребителите в азбучен ред. Подредете IPтата в азбучен ред. В нашия пример, изходът би би следния:

- `alex: 62 [10.10.17.33, 212.50.118.81]`
- `peter: 303 [10.10.17.34, 10.10.17.35, 192.168.0.11]`

Вход

Входните данни се въвеждат в конзолата. На първи ред ще получите числото n, което ще ви покаже колко реда от лога ще получите. Всеки от следващите n реда ще пази информация от лога във формат <IP> <потребител> <продължителност>. Входните данни винаги ще са валидни и ще отговарят на поосечения формат. Няма нужда да ги проверявате.

Изход

Изведете по един ред за всеки потребител (подреждайте потребителите в азбучен ред). За всеки потребител извеждайте сумата от продължителността и всички негови IPта, подредени азбучно във формат <потребител>: <продължителност> [<IP_1>, <IP_2>, ...]. Премахнете всякакви дублирани стойности в IP адресите и ги подредете азбучно (както се подреждат низове).

Ограничения

- Броят n на редовете е в интервала [1...1000].
- <IP> е стандартен IP адрес във формата a.b.c.d където a, b, c и d са цели числа в интервала [0...255].
- <потребителско име> се състои от латински букви, с дължина [1...20].
- <продължителност> е цяло число в интервала [1...1000].
- Лимит по време: 0.3 sec. Лимит по памет: 16 MB.

Примери

Вход	Изход
7 192.168.0.11 peter 33 10.10.17.33 alex 12 10.10.17.35 peter 30 10.10.17.34 peter 120 10.10.17.34 peter 120 212.50.118.81 alex 46 212.50.118.81 alex 4	alex: 62 [10.10.17.33, 212.50.118.81] peter: 303 [10.10.17.34, 10.10.17.35, 192.168.0.11]
2 84.238.140.178 nakov 25	nakov: 60 [84.238.140.178]

84.238.140.178 nakov 35	

Решение

```
namespace LogsProcessing
{
    public class Program
    {
        static void Main(string[] args)
        {
            Dictionary<string, int> users = new Dictionary<string, int>();
            Dictionary<string, List<string>> ips = new Dictionary<string,
List<string>>();

            var n = int.Parse(Console.ReadLine());

            for (int i = 0; i < n; i++)
            {
                var log = Console.ReadLine().Split().ToArray();

                var ip = log[0];
                var user = log[1];
                var time = int.Parse(log[2]);

                if (users.ContainsKey(user))
                {
                    users[user] += time;
                    var list = ips[user];
                    if (!list.Contains(ip)) list.Add(ip);
                    ips[user] = list;
                }
                else
                {
                    users.Add(user, time);
                    var list = new List<string>();
                    list.Add(ip);
                    ips.Add(user, list);
                }
            }

            foreach (var user in users.OrderBy(x => x.Key))
            {
                Console.WriteLine($"{user.Key}: {user.Value} [{string.Join(", ",
ips[user.Key])}]");
            }
        }
    }
}
```

Тема 8. Подготовка за практически изпит

Problem 8.1. Resurrection

You ever heard of Phoenixes? Magical Fire Birds that are practically immortal – they reincarnate from an egg when they die. Naturally, it takes time for them to

reincarnate. You will play the role of a scientist who calculates the time to reincarnate for each phoenix, based on its body parameters.

You will receive N, an integer – the amount of phoenixes.
For each phoenix, you will receive 3 input lines:

- On the first input line you will receive an integer – the total length of the body of the phoenix.
- On the second input line you will receive a floating-point number – the total width of the body of the phoenix.
- On the third input line you will receive an integer – the length of 1 wing of the phoenix.

For each phoenix, you must print the years it will take for it to reincarnate, which is calculated by the following formula:

The **totalLength** powered by **2**, multiplied by the sum of the **totalWidth** and the **totalWingLength** (**2 * wingLength**).

totalYears = {totalLength} ^ 2 * ({totalWidth} + 2 * {wingLength})

Input

- On the first input line you will receive N, an integer – the amount of phoenixes.
- On the next N * 3 input lines you will be receiving data for each phoenix.

Output

- As output, you must print the total years needed for reincarnation for each phoenix.
- Print each phoenix's years when you've calculated them.
- Print each phoenix's years on a new line.

Constrains

- The amount of phoenixes will be an integer in range [0, 1000].
- The total length of the body of the phoenix will be an integer in range [-2^{31}, 2^{31}].
- The total width of the body of the phoenix will be a floating-point number in range [-2^{31}, 2^{31}].
- The total width of the body of the phoenix will have up to 20 digits after the decimal point.
- The total length of the wing of the phoenix will be an integer in range [-2^{31}, $2^{31} - 1$].
- The total years is a product of integers and floating-point numbers, thus it is a floating-point number.
- The total years should have the same accuracy as the total width.
- Allowed working time / memory: 100ms / 16MB.

Examples

Input	Output	Comments

2 100 50 30 150 25 10	1100000 1012500	2 phoenixes: P1: Body length: 100 Body width: 50 Length of 1 wing: 30 Total years: 100 ^ 2 * (50 + 2 * 30) = 1100000 P2: Body length: 150 Body width: 25 Length of 1 wing: 10 Total years: 150 ^ 2 * (25 + 2 * 10) = 1012500
2 100 50.243 31 154 23.132 11	1122430.000 1070350.512	2 phoenixes: P1: Body length: 100 Body width: 50.243 Length of 1 wing: 31 Total years: 100 ^ 2 * (50.243 + 2 * 31) = 1122430.000 P2: Body length: 154 Body width: 23.132 Length of 1 wing: 11 Total years: 154 ^ 2 * (23.132 + 2 * 11) = 1070350.512

Solution

```
namespace Resurrection
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            while (n > 0)
            {
                int totalLength = int.Parse(Console.ReadLine());
                float totalWidth = float.Parse(Console.ReadLine());
                int wingLength = int.Parse(Console.ReadLine());

                decimal totalYears = (decimal)((totalLength * totalLength) *
(totalWidth + (2 * wingLength)));
                Console.WriteLine(totalYears);
                n--;
            }
        }
```

```
    }
}
```

Problem 8.2. Icarus

Icarus is the majestic phoenix who has been alive from the beginning of creation. Icarus travels through different planes. When Icarus travels through a plane, he damages Reality itself with his overwhelming, beyond godlike flames.

You will receive a sequence of integers – the plane. After that you will receive 1 integer – an index in that sequence, which is Icarus's starting position. Icarus's INITIAL DAMAGE is 1.

You will then begin receiving commands in the following format: "**{direction} {steps}**". The direction will be either "**left**" or "**right**", and the steps will be an integer. Depending on the direction, Icarus must step through the sequence of integers to the left or right. Each time he steps on a NEW position, he damages it. In other words, he SUBTRACTS his current damage from the integer at that position. Walking left and right has its conditions though:

- If Icarus passes beyond the start of the sequence (`index: -1`) while going left, he must go at the end of the sequence (`index: length – 1`).
- If Icarus passes beyond the end of the sequence (`index: length - 1`) while going right, he must go at the start of the sequence (`index: 0`).

If 1 of the 2 cases stated above happens, Icarus increments his damage by 1.

The input ends when you receive the command "Supernova". When that happens you must print what is left of the sequence.

Input

- On the first input line you will get the sequence of integers, separated by spaces.
- On the second input line you will get Icarus's starting position.
- On the next several input lines you will get the commands.

Output

- As output you must print a single line containing the remaining elements of the sequence, separated by spaces.

Constrains

- The integers in the sequence will be in range [0, 1000].
- The initial position of Icarus will always be valid and inside the sequence's indexes.
- The direction will always be either "`left`" or "`right`".
- The steps will be in range [0, 1000].
- There will be NO invalid input lines.
- Allowed working time / memory: 100ms / 16MB.

Examples

Input	Output	Comments
50 50 25 50 50 3 left 2 right 2 left 2 right 2 Supernova	50 48 21 48 50	Initial index: 3 Initial state: 50 50 25 50 50 Go left 2 steps: 50 50 24 50 50 50 49 24 50 50 Go right 2 steps: 50 49 23 50 50 50 49 23 49 50 Go left 2 steps: 50 49 22 49 50 50 48 22 49 50 Go right 2 steps: 50 48 21 49 50 50 48 21 48 50 Final state: 50 48 21 48 50
5 3 5 5 5 2 left 5 left 5 Supernova	2 0 0 0 0	Initial index: 2 Initial state: 5 3 5 5 5 Go left 5 steps: 5 2 5 5 5 4 2 5 5 5 4 2 5 5 3 4 2 5 3 3 4 2 3 3 3 Go left 5 steps: 4 0 3 3 3 2 0 3 3 3 2 0 3 3 0 2 0 3 0 0 2 0 0 0 0 Final state: 2 0 0 0 0

Solution

```
namespace Icarus
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            var nums = Console.ReadLine().Split().Select(int.Parse).ToArray();
            int start = int.Parse(Console.ReadLine());
            int pos = start, damage = 1;
            var command = Console.ReadLine().Split().ToArray();
            do
            {
                int steps = int.Parse(command[1]);
                if (command[0] == "left")
                {
                    while (steps > 0)
                    {
                        pos--;
                        if (pos < 0)
                        {
                            pos = nums.Length - 1;
                            damage++;
                        }
                        nums[pos] -= damage;
                        steps--;
                    }
                }
                if (command[0] == "right")
                {
                    while (steps > 0)
                    {
                        pos++;
                        if (pos > nums.Length - 1)
                        {
                            pos = 0;
                            damage++;
                        }
                        nums[pos] -= damage;
                        steps--;
                    }
                }
                command = Console.ReadLine().Split().ToArray();
            }
            while (command[0] != "Supernova");
            Console.WriteLine(String.Join(" ", nums));
        }
    }
}
```

Problem 8.3. Phoenix Grid

The Phoenix Grid is an ancient artifact created by the Linguistics miracle – Mozilla, The "Fire Bird". It is used to translate Phoenix language. You are the newest scientist, researching the Grid and as the research team was almost out of hope, you came up with the genius idea to use Regular Expressions! You saved the day! You are a Hero!

You will begin receiving encoded messages. You must CHECK each one of them and if it's a VALID.

A valid encoded message consists of one phrase or more phrases, separated by DOTS ('.').

- A phrase consists of exactly 3 characters.
- A phrase CANNOT contain whitespace characters or the '_' (**underscore**) character.

Valid messages: "**asd.dsa**", "**123.312**", "**3@a.231**", "**111**", "**@sd**", "**132.31$.ddd**" . . .

Invalid messages: "**123asdasd.dsa**", "**_@a. sd**", "**a.s.d**" . . .

When you have found a valid message, you must check if it a PALINDROME – if it reads the same backward as forward.

Palindrome messages: "**asd.dsa**", "**123.321**", "**cat.php.tac**" . . .

If the message is VALID and is a PALINDROME print "YES". In any other case, print "NO".

The input ends when you receive the command "**ReadMe**".

Input

- As input you will receive several input lines containing encoded messages.

Output

- As output you must print for each message "YES" or "NO" if its valid or not.

Constrains

- The input lines may contain any ASCII character.
- There will be no more than 1000 input lines.
- Allowed working time / memory: 100ms / 16MB.

Examples

Input	Output
asd asd.asd asd.dsa 123.323.321 _ds._sad.sds jss.csh.php.hsc.ssj ReadMe	NO NO YES YES NO YES
asa igi.igi ___.___ . sds.dsd.sds.dsd.sds.dsd.sds	YES YES NO NO YES

xha.ahx ReadMe	YES

Solution

```
namespace PhoenixGrid
{
    class Program
    {
        static void Main(string[] args)
        {
            string message = Console.ReadLine();

            while (message != "ReadMe")
            {
                bool IsAPalindrome = true;

                for (int i = 3; i <= message.Length - 1; i += 4)
                {
                    if (message[i] != '.')
                    {
                        IsAPalindrome = false;
                    }
                }

                if (message.Contains(' ') || message.Contains('_') ||
message.Contains('\t'))
                {
                    IsAPalindrome = false;
                }

                for (int i = 0; i <= message.Length / 2; i++)
                {
                    if (message[i] != message[message.Length - 1 - i])
                    {
                        IsAPalindrome = false;
                    }
                }

                if (IsAPalindrome == true)
                {
                    Console.WriteLine("YES");
                }
                else
                {
                    Console.WriteLine("NO");
                }

                message = Console.ReadLine();
            }
        }
```

```
    }
}
```

Problem 8.4. Phoenix Oscar Romeo November

The fire creatures are assembling in squads to fight The Evil Phoenix God. You have been tasked to determine which squad is the strongest, so it will be sent as The Vanguard.

You will begin receiving input lines containing information about fire creatures in the following format:

{creature} -> {squadMate}
The **creature** and the **squadMate** are strings. You should store every creature, and his squad mates. If the creature already exists, you should add the new squad mate to it.

- If there is already a squad mate with the given name in the given creature's squad, IGNORE that line of input.
- If the given squad mate name is the same as the given creature, IGNORE that line of input.

The input sequence ends when you receive the command **"Blaze it!"**.

When that happens you must print the creatures ordered in descending order by count of squad mates. Sounds simple right? But there is one little DETAIL.

If a particular **creature** has a **squadMate**, and that **squadMate** has that **creature** in his **squadMates**, you should NOT consider them as part of the count of squad mates.

Example:

Creature 1: Mozilla -> {Tony, Dony, Mony}

Creature 2: Tony -> {Mozilla, Franzilla, Godzilla}

Mozilla has 2 squad mates in total, because Tony also has Mozilla in his squad mates.

Tony has 2 squad mates in total, because Mozilla also has Tony in his squad mates.

Input

- **As input you will receive several input lines containing information about the fire creatures.**
- **The input sequence ends when you receive the command "`Blaze it!`".**

Output

- As output you must print each of the creatures the following information:
    - {creature} : {countOfSquadMates}

- As it was stated above, mind the count of squad mates. If 2 creatures have themselves in their squad mates, they should NOT be counted.

Constrains

- The `creature` and the `squadMate` will be strings which may contain any ASCII character.
- There will be NO invalid input lines.
- Allowed time / memory: 100ms / 16MB.

Examples

Input	Output
Mozilla -> Tony Tony -> Godzilla Mozilla -> Dony Tony -> Franzilla Mozilla -> Mony Tony -> Mozilla Blaze it!	Mozilla : 2 Tony : 2
FireBird -> FireMane Phoenix -> FireVoid FireVoid -> FireMane FireSnow -> FireMane Phoenix -> FireBird FireMane -> FireBird FireMane -> FireVoid Phoenix -> FireSnow FireMane -> FireSnow FireMane -> FireMane Phoenix -> FireMane Phoenix -> FireVoid Blaze it!	Phoenix : 4 FireBird : 0 FireVoid : 0 FireSnow : 0 FireMane : 0

Solution

```
namespace PhoenixOscarRomeoNovember
{
    class Program
    {
        static void Main(string[] args)
        {
            var dict = new Dictionary<String, List<String>>();

            string line = Console.ReadLine();
            while (line != "Blaze it!")
            {
                var Parts = line.Split('>');
                var Creature = Parts[0].Trim('-').Trim(' ');
                var SquadMate = Parts[1].Trim(' ');
                if (Creature != SquadMate)
```

```
            {
                if (!dict.ContainsKey(Creature))
                {
                    dict.Add(Creature, new List<String>() { SquadMate });
                }
                else if (!dict[Creature].Contains(SquadMate))
                {
                    dict[Creature].Add(SquadMate);
                }
            }
            line = Console.ReadLine();
        }

        var nope = new Dictionary<String, List<String>>();
        foreach (var creature in dict)
        {
            var list = new List<String>();
            foreach (var mate in creature.Value)
            {
                if (dict.ContainsKey(mate))
                {
                    if (dict[mate].Contains(creature.Key))
                    {
                        continue;
                    }
                }
                list.Add(mate);
            }
            nope.Add(creature.Key, list);
        }

        var sorted = nope.OrderByDescending(x => x.Value.Count);
        foreach (var item in sorted)
        {
            Console.WriteLine("{0} : {1}", item.Key, item.Value.Count);
        }
    }
  }
}
```

Problem 8.5. Anonymous Downsite

The Anonymous informal group of activists have hacked a few commercial websites and the CIA has hired you to write a software which calculates the losses. Based on the given data, use the appropiate data types.

You will receive 2 input lines – each containing an integer.

- The first is N – the number of websites which are down.
- The second is the **`security key`**.

On the next N lines you will receive data about websites in the following format: **{siteName} {siteVisits} {siteCommercialPricePerVisit}**

You must calculate the **site loss** by the following formula: **siteVisits * siteCommercialPricePerVisit**

When you finish reading all data, you must print the affected sites' names – each on a new line.
Then you must print the **total money loss** – sum of all **site loss**, on a new line.
Finally you must print the **security token**, which is the **security key**, POWERED by the COUNT of affected sites.

Input

- On the first input line you will get N – the count of affected websites.
- On the second input line you will the security key.
- On the next N input lines you will get data about the websites.

Output

- As output you must print all affected websites' names – each on a new line.
- After the website names you must print the total loss of data, printed to the 20th digit after the decimal point. The format is "`Total Loss: {totalLoss}`".
- Finally you must print the security token. The format is "`Security Token: {securityToken}`".

Constrains

- The integer N will be in range [0, 100].
- The security token will be in range [0, 10].
- The website name may contain any ASCII character except whitespace.
- The site visits will be an integer in range [0, 2^{31}].
- The price per visit will be a floating point number in range [0, 100] and will have up to 20 digits after the decimal point.
- Allowed working time/memory: 100ms / 16MB.

Examples

Input	Output
3 8 www.google.com 122300 94.23233 www.abv.bg 2333 11 www.kefche.com 12322 23.3222	www.google.com www.abv.bg www.kefche.com Total Loss: 11837653.10740000000000000000 Security Token: 512
1 1 www.facebook.com 100000 10.45	www.facebook.com Total Loss: 1045000.00000000000000000000 Security Token: 1

Solution

```
namespace AnonymousDownsite
{
```

```
    class Program
    {
        static void Main(string[] args)
        {
            var downWebsites = int.Parse(Console.ReadLine());
            var securityKey = int.Parse(Console.ReadLine());

            decimal totalLost = 0;
            var securityToken = Math.Pow(securityKey, downWebsites);
            var websites = new List<String>();

            while (downWebsites > 0)
            {
                var line = Console.ReadLine().Split().ToArray();

                websites.Add(line[0]);
                var siteVisits = int.Parse(line[1]);
                var siteCommercialPricePerVisit = decimal.Parse(line[2]);

                var siteLoss = siteVisits * siteCommercialPricePerVisit;
                totalLost += siteLoss;
                downWebsites--;
            }

            Console.WriteLine(String.Join("\n", websites));
            Console.WriteLine("Total Loss: {0:f20}", totalLost);
            Console.WriteLine("Security Token: {0}", securityToken);
        }
    }
}
```

Problem 8.6. Anonymous Threat

The Anonymous have created a cyber hypervirus which steals data from the CIA. You, as the lead security developer in CIA, have been tasked to analyze the software of the virus and observe its actions on the data. The virus is known for his innovative and unbeleivably clever technique of merging and dividing data into partitions.

You will receive a single input line containing STRINGS separated by spaces. The strings may contain any ASCII character except whitespace.

You will then begin receiving commands in one of the following formats:

- merge {startIndex} {endIndex}
- divide {index} {partitions}

Every time you receive the **merge command**, you must merge all elements from the **startIndex**, till the **endIndex**. In other words, you should concatenate them. Example: **{abc, def, ghi} -> merge 0 1 -> {abcdef, ghi}**

If any of the given indexes is out of the array, you must take ONLY the range that is INSIDE the array and merge it.

Every time you receive the **divide command**, you must DIVIDE the element at the given index, into several small substrings with equal length. The count of the substrings should be equal to the given partitions.

Example: **{abcdef, ghi, jkl} -> divide 0 3 -> {ab, cd, ef, ghi, jkl}**

If the string CANNOT be exactly divided into the given partitions, make all partitions except the LAST with EQUAL LENGTHS, and make the LAST one – the LONGEST.

Example: **{abcd, efgh, ijkl} -> divide 0 3 -> {a, b, cd, efgh, ijkl}**

The input ends when you receive the command **"3:1"**. At that point you must print the resulting elements, joined by a space.

Input

- The first input line will contain the array of data.
- On the next several input lines you will receive commands in the format specified above.
- The input ends when you receive the command "`3:1`".

Output

- As output you must print a single line containing the elements of the array, joined by a space.

Constrains

- The strings in the array may contain any ASCII character except whitespace.
- The `startIndex` and the `endIndex` will be in range [-1000, 1000].
- The `endIndex` will ALWAYS be GREATER than the `startIndex`.
- The `index` in the `divide command` will ALWAYS be INSIDE the array.
- The `partitions` will be in range [0, 100].
- Allowed working time/memory: 100ms / 16MB.

Examples

Input	Output

`Ivo Johny Tony Bony Mony`
`merge 0 3`
`merge 3 4`
`merge 0 3`
`3:1`
	`IvoJohnyTonyBonyMony`

`abcd efgh ijkl mnop qrst uvwx yz`
`merge 4 10`
`divide 4 5`
`3:1`
	`abcd efgh ijkl mnop qr st uv wx yz`

Solution

```
namespace AnonymousThreat
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            var names = Console.ReadLine().Split().ToList();
            var line = Console.ReadLine().Split();

            while (line[0] != "3:1")
            {
                var command = line[0];

                if (command == "merge")
                {
                    var start = int.Parse(line[1]);
                    var end = int.Parse(line[2]);
                    var denyStart = start < 0;
                    var denyEnd = end >= names.Count;
                    if (denyStart) start = 0;
                    if (denyEnd) end = names.Count - 1;
                    for (int i = start; i < end; end--)
                    {
                        names[i] = names[i] + names[i + 1];
                        names.RemoveAt(i + 1);
                    }
                }

                if (command == "divide")
                {
                    var index = int.Parse(line[1]);
                    var partitions = int.Parse(line[2]);
                    string currentString = names[index];
                    var lenghtOfPartitions = currentString.Length / partitions;
                    var additions = new List<string>(partitions);
                    for (int i = 0; i < partitions - 1; i++)
                    {
                        string currentAdition = currentString.Substring(0,
lenghtOfPartitions);
                        additions.Add(currentAdition);
                        currentString = currentString.Substring(lenghtOfPartitions);
                    }
                    additions.Add(currentString);
                    names.RemoveAt(index);
                    names.InsertRange(index, additions);
                }

                line = Console.ReadLine().Split();
            }
            Console.WriteLine(String.Join(" ", names));
        }
    }
}
```

Problem 8.7. Anonymous Vox

The Anonymous's main communication channel is based on encoded messages. The CIA has targetted that channel, assuming that it holds sensitive information. You have been hired to decode and break their internal com. system.

You will receive an input line containing a single string – the encoded text. Then, on the next line you will receive several values in the following format: "**{value1}{value2}{value3}...**".

You must find the encoded placeholders in the text and REPLACE each one of them with the value that corresponds to its index.

Example: placeholder1 – value1, placeholder2 – value2 etc. There may be more values than placeholders or more placeholders than values.

The placeholders consist of 3 blocks **{start}{placeholder}{end}**. The start should consist only of English alphabet letters. The placeholder may contain ANY ASCII character. The end should be EXACTLY EQUAL to the start. The idea is that you have to find the placeholders, and REPLACE their **placeholder** block with the value at that index.

Example Placeholders: "**a.....a**", "**b!d!b**", "**asdxxxxxasd**", "**peshogoshopesho**"...

You must ALWAYS match the placeholder with the LONGEST **start** and the RIGHTMOST **end**. For example if you have "**asddvdasd**" you should NOT match "**dvd**" as a placeholder, you should match "**asddvdasd**".

At the end you must print the result text, after you've replaced the values.

Input

- On the first input line you will receive the encoded text.
- On the second input line you will receive the placeholders.

Output

- As output you must print a single line containing the resulting text, after the replacing of values.

Constrains

- The given text may contain ANY ASCII character.
- The given values may contain ANY ASCII character except '{' and '}'.
- The given values will AWLAYS follow the format specified above.
- Allowed working time/memory: 100ms / 16MB.

Examples

Input	Output
Hello_mister,_Hello { Jack }	Hello Jack Hello

ASD___asdfffasd {this}{exam}{problem}{is}{boring}	ASD___asdthisasd
Whatsup_ddd_sup {Dude}	WhatsupDudesup
HeypalHey_____how_ya_how_doin_how {first}{second}	HeyfirstHey_____howsecondhow

Solution

```
using System.Text.RegularExpressions;

namespace AnonymousVox
{
    class Program
    {
        static void Main(string[] args)
        {
            var placeholderRegex = new Regex(@"([A-Za-z]+).+\1");
            var valueRegex = new Regex(@"{(.+?)}");

            string text = Console.ReadLine();
            string valuesText = Console.ReadLine();

            var values = valueRegex.Matches(valuesText);
            int i = 0;
            text = placeholderRegex.Replace(text, m =>
            {
                if (i < values.Count)
                {
                    return
$"{m.Groups[1].Value}{values[i++].Groups[1].Value}{m.Groups[1].Value}";
                }
                return m.Value;
            });

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

Problem 8.8. Anonymous Cache

The Anonymous are storing data on their dataservers about their activities. The CIA has higher the greatest hacker in the world – You. Your job is to extract their data and send it to the CIA. It won't be an easy task, Get Ready!

You will receive several input lines in one of the following formats:

- {dataSet}
- {dataKey} -> {dataSize} | {dataSet}

The **dataSet** and **dataKey** are both strings. The **dataSize** is an integer. The **dataSets** hold **dataKeys** and their **dataSizes**.

If you receive only a **dataSet** you should add it. If you receive a **dataKey** and a **dataSize**, you should add them to the given **dataSet**.

And here's where the fun begins. If you receive a **dataKey** and a **dataSize**, but the given **dataSet** does NOT exist, you should STORE those keys and values in a **cache**. When the corresponding **dataSet** is added, you should check if the **cache** holds any keys and values referenced to it, and you should add them to the **dataSet**.

You should end your program when you receive the command "**thetinggoesskrra**". At that point you should extract the **dataSet** from the **data** with the HIGHEST **dataSize** (SUM of all its **dataSizes**), and you should print it.

NOTE: Elements in the **cache**, should be CONSIDERED NON-EXISTANT. You should NOT count them in the final output.

In case there are NO **dataSets** in the **data**, you should NOT do anything.

Input

- The input comes in the form of commands in one of the formats specified above.
- The input ends when you receive the command "`thetinggoesskrra`".

Output

- As output you must print the `dataSet` with the HIGHEST SUM of all `dataSizes`.
- The output format is:

  Data Set: {dataSet}, Total Size: {sumOfAllDataSizes}

  $.{dataKey1}

  $.{dataKey2}

  ...
- In case there are NO `dataSets` in the `data`, print nothing.

Constrains

- The `dataSet` and `dataKey` are both strings which may contain ANY ASCII character except ' ', '-', '>', '|'.
- The `dataSize` is a valid integer in range [0, 1.000.000.000].
- There will be NO invalid input lines.
- There will be NO `dataSets` with EQUAL SUMMED `dataSize`.
- There will be NO DUPLICATE keys.
- Allowed working time/memory: 100ms / 16MB.

Examples

Input	Output

Users BankAccounts ADDB444 -> 23111 | BankAccounts Students -> 2000 | Users Workers -> 24233 | Users thetinggoesskrra	Data Set: Users, Total Size: 26233 $.Students $.Workers
Cars Car1 -> 233333 | Cars Car23 -> 266666 | Cars Warehouse2 -> 10000 | Buildings Warehouse3 -> 480000 | Buildings Warehouse5 -> 100000 | Buildings Buildings thetinggoesskrra	Data Set: Buildings, Total Size: 590000 $.Warehouse2 $.Warehouse3 $.Warehouse5

Solution

```
namespace AnonymousCache
{
    class Program
    {
        static void Main(string[] args)
        {
            var data = new Dictionary<String, Dictionary<String, long>>();
            var cache = new Dictionary<String, Dictionary<String, long>>();

            string line = Console.ReadLine();
            while (line != "thetinggoesskrra")
            {
                var parts = line.Replace(" -> ", ";").Replace(" | ",
";").Split(';');
                if (parts.Count() > 1)
                {
                    String dataKey = parts[0];
                    long dataSize = long.Parse(parts[1]);
                    String dataSet = parts[2];

                    if (data.ContainsKey(dataSet))
                    {
                        data[dataSet].Add(dataKey, dataSize);
                    }
                    else if (!cache.ContainsKey(dataSet))
                    {
                        cache.Add(dataSet, new Dictionary<string, long>() { {
dataKey, dataSize } });
                    }
                    else
                    {
                        cache[dataSet].Add(dataKey, dataSize);
                    }

                }
                else if (parts[0] != "thetinggoesskrra")
```

```
                {
                    String dataSet = parts[0];
                    if (cache.ContainsKey(dataSet))
                    {
                        data.Add(dataSet, cache[dataSet]);
                        cache.Remove(dataSet);
                    }
                    else
                    {
                        data.Add(dataSet, new Dictionary<string, long>());
                    }
                }
                line = Console.ReadLine();
            }

            long max = data.Values.Max(x => x.Values.Sum());

            foreach (var dataSet in data)
            {
                long sum = dataSet.Value.Sum(x => x.Value);
                if (sum == max)
                {
                    Console.WriteLine("Data Set: {0}, Total Size: {1}", dataSet.Key,
sum);
                    foreach (var dataKey in dataSet.Value)
                    {
                        Console.WriteLine("$.{0}", dataKey.Key);
                    }
                }
            }
        }
    }
}
```

Problem 8.9. Raindrops

The Raindear Forecast Agency (RFA) is an organization founded by an old and kind grandma which wanted quality forecasts. The Agency has hired you to write a software which finds the Rain Coefficient, by calculating simple input data.

You will receive N, an integer – the amount of regions. Then you will receive the density – a floating-point number.

For each region, you will receive an input line in the following format:

"{raindropsCount} {squareMeters}"

The **raindropsCount** and the **squareMeters** will be integers. Your task is to calculate the regional coefficient by the following formula: **raindropsCount / squareMeters**

NOTE: The regional coefficient should be a floating-point number.

Your task is to sum all regional coefficients, and then divide it by the **density**, and print the result.

If a division is not possible, just print the sum of all regional coefficients.

Input

- On the first input line you will receive N – the amount of regions.
- On the second input line you will receive the density.
- On the next N input lines you will receive information about the regions.

Output

- As output you must print the sum of all regional coefficients divided by the density.
- If a division is not possible you must print the sum of all regional coefficients.
- The output should be rounded and printed to 3 places after the decimal point.

Constraints

- The amount of regions – N will be an integer in range [0, 100].
- The density will be a floating-point number in range [0, 9].
- The raindropsCount will be an integer in range [-2^{31}, 2^{31}].
- The squareMeters will be an integer in range [1, 10000].
- Allowed working time / memory: 100ms / 16MB.

Examples

Input	Output	Comment
4 4 2000 10 1000 5 5000 2000 3000 30	125.625	2000 / 10 = 200 1000 / 5 = 200 5000 / 2000 = 2.5 3000 / 30 = 100 200 + 200 + 2.5 + 100 = 502.5 502.5 / 4 = 125.625
2 2 100000 50 200000 25	5000.000	100000 / 50 = 2000 200000 / 25 = 8000 2000 + 8000 = 10000 10000 / 2 = 5000 (rounded till 3rd symbol) = 5000.000

Solution

```
namespace Raindrops
{
    class Program
    {
        static void Main(string[] args)
        {
            var regions = int.Parse(Console.ReadLine());
            var density = float.Parse(Console.ReadLine());
            var regionalCoefficient = new List<float>();
```

```
            while (regions > 0)
            {
                var line = Console.ReadLine().Split().Select(float.Parse).ToArray();
                var raindropsCount = line[0];
                var squareMeters = line[1];
                regionalCoefficient.Add(raindropsCount / squareMeters);
                regions--;
            }

            var sum = regionalCoefficient.Sum();
            Console.WriteLine("{0:f3}", sum / density);
        }
    }
}
```

Problem 8.10. Rainer

A Rainer is like a runner but in Rain. One who runs from the Rain. Donald is one good Rainer and he created a game where he dodges raindrops at lightning fast speed through some incomprehensible logic.

You will receive a sequence of integers – each of those integers, except the last one, form the game field.

You must take the last integer from that sequence – that is the initial index at which Donald steps.

The game goes so – you must decrease all of the integers in the sequence' values by 1.

Then you must read an integer – the next index at which Donald steps.

You must repeat these steps until Donald gets wet.

If an integer reaches 0, that means a raindrop has fallen there. If Donald is on that position, he gets wet.

If an integer reaches 0, and Donald is not there, you must return the integer to its original value. (initial value)

When Donald gets wet, the program ends, and you must print the current sequence of integers, and the count of steps Donald has made (the initial index does not count as a step)

Input

- On the first input line you will get the sequence of integers, separated by spaces.
- On the next several input lines you will be getting integers – the indexes.

Output

- As output you must print the sequence of integers, separated by spaces, on one line.
- Then you must print the steps Donald has made on the second line.

Constraints

- The count of the integers in the sequence will be [3, 100].
- The integers in the sequence will be in range [2, 100].
- The indexes that will be given to you will always be valid and inside the sequence.
- Allowed working time / memory: 100ms / 16MB.

Examples

Input	Output	Comment
5 2 3 4 5 3 0 1 4 1 1	4 0 0 2 4 5	Sequence – 5 2 3 4 5, Initial Index – 3 We decrease all by 1, Sequence – 4 1 2 3 4 We check if Donald is on an element 0. He is not, so we read next step. Index – 0. Steps – 1. Sequence – 3 0 1 2 3. There is an element with value 0, but Donald is not there, we return it to its original value (2). Sequence – 3 2 1 2 3. Index – 1. Steps – 2. Sequence – 2 1 3 1 2. Index – 4. Steps – 3. Sequence – 1 2 2 4 1. Index – 1. Steps – 4. Sequence – 5 1 1 3 5. Index – 1. Steps – 5. We decrease by 1, and it gets 4 0 0 2 4. Donald is on Index 1 – which is currently 0. He dies. No other steps are made, and the program ends.
2 3 4 5 6 2 1 2 3 4 0	0 0 2 4 0 5	

Solution

```
namespace Rainer
{
    class Program
    {
        static void Main(string[] args)
        {
            var input = Console.ReadLine().Split().Select(int.Parse).ToList();
            input.RemoveAt(input.Count - 1);
            List<int> original = new List<int>();
            var steps = 0;
            foreach (var re in input)
            {
                original.Add(re);
```

```
            }
            int indexnow = input.Last();
            for (int i = 0; i < input.Count; i++)
            {
                input[i]--;
            }
            while (true)
            {
                int a = int.Parse(Console.ReadLine());
                steps++;
                for (int i = 0; i < input.Count; i++)
                {
                    input[i]--;
                }
                if (input[a] == 0)
                {
                    Console.WriteLine(string.Join(" ", input));
                    Console.WriteLine(steps);
                    break;
                }
                for (int i = 0; i < input.Count; i++)
                {
                    if (input[i] == 0)
                    {
                        input[i] = original[i];
                    }
                }
            }
        }
    }
}
```

Problem 8.11. Raincast

The Raindear Forecast Agency has hired you again, astonished by your previous works. This time you are hired to write a software which receives Telegram Raincasts, and validates them. The messages are quite scrambled so you only have to find the valid ones.

You will begin receiving input lines which may contain any ASCII character. Your task is to find the Raincasts.

The Valid Raincast consists of 3 lines:

- Type: {type}
- Source: {source}
- Forecast: {forecast}

The **type** should either be "**Normal**", "**Warning**" or "**Danger**".

The **source** should consist of alphanumeric characters.

The **forecast** should not contain any of the following characters: '**!**', '**.**', '**,**', '**?**'.

- When you find a **type**, you must search for a **source**.
- When you find a **source** you must search for a **forecast**.
- When you find a **forecast**, you have completed a single Valid Raincast. You must start searching for a **type** again, for the next Raincast.

There might be invalid lines between the valid ones. You should keep the order of searching.

NOTE: The valid input lines must be exactly in the format specified above. Any difference makes them invalid.

When you receive the command "**Davai Emo**", the input ends. You must print all valid raincasts you've found, each in a specific format, each on a new line.

Input

- The input will come in several input lines which may contain any ASCII character.
- The input ends when you receive the command "`Davai Emo`".

Output

- As output you must print all of the valid raincasts you've found, each on a new line.
- The format is: `({type}) {forecast} ~ {source}`

Constraints

- The input lines may contain any ASCII character.
- There will be no more than 100 input lines.
- Allowed working time / memory: 100ms / 16MB.

Examples

Input	Output
Type: Normal Source: JohnKutchur9 Forecast: A full rain program no sun Type: Danger Forecast: Invalid Input Line Source: IvoAndreev Type: Invalid Input Line Forecast: Shte vali qko Davai Emo	(Normal) A full rain program no sun ~ JohnKutchur9 (Danger) Shte vali qko ~ IvoAndreev
Forecast: Bau Source: Myau Type: Strong Source: Good	(Warning) Nqma da se kefim mn na praznici ~ Emo

Forecast: Valid Type: Warning Type: Danger Source: Emo Forecast: Nqma da se kefim mn na praznici Davai Emo	

Solution

```
namespace Raincast
{
    class Program
    {
        static void Main(string[] args)
        {
            String line = Console.ReadLine();
            String[] ValidForecast = new String[3];
            while (line != "Davai Emo")
            {
                var parts = line.Split(':');
                switch (parts[0])
                {
                    case "Type":
                        {
                            var type = parts[1].Trim(' ');
                            if (type == "Normal" || type == "Warning" || type ==
"Danger" &&
                                 String.IsNullOrEmpty(ValidForecast[0]))
                            {
                                ValidForecast[0] = type;
                            }
                            break;
                        }
                    case "Source":
                        {
                            var src = parts[1].Trim(' ');
                            if (!String.IsNullOrEmpty(ValidForecast[0]))
                            {
                                ValidForecast[1] = src;
                            }
                            break;
                        }
                    case "Forecast":
                        {
                            var frc = parts[1].Trim(' ');
                            if (!String.IsNullOrEmpty(ValidForecast[0]) &&
                                !String.IsNullOrEmpty(ValidForecast[1]))
                            {
                                ValidForecast[2] = frc;
                                Console.WriteLine("({0}) {1} ~ {2}",
ValidForecast[0], ValidForecast[2], ValidForecast[1]);
                                ValidForecast = new String[3];
                            }
```

```
                        break;
                    }
                }
                line = Console.ReadLine();
            }
        }
    }
}
```

Problem 8.12. RainAir

Before naming it RyanAir … Tony Ryan named it RainAir, because the day he named it, it was really rainy, and he liked rain. Anyways, you have been hired by Tony, to create a software which manipulates data about flights and customers. The future of RyanAir is in your hands.

You will receive input lines in one of the following formats:

- {customerName} {customerFlight1} {customerFlight2} {customerFlight3} …
- {customerName} = {customer2Name}

The **customerName** is a string. The **customerFlights** are integers.

If you receive a **customerName** and **customerFlights**, you should add the customer and the flights to the customer.

If the customer already exists, just add the new flights to him.

If you receive a **customerName** and **customer2Name**, you should make the 1st customer's flights equal to the 2nd customer's flights.

The input ends when you receive the command "**I believe I can fly!**". When that happens, you must print all customers, ordered by count of flights in descending order, and then by alphabetical order.

The flights must be ordered in ascending order.

Input

- The input consists of several input lines in the format specified above.
- The input ends when you receive the command "`I believe I can fly!`".

Output

- As output you must print all the customers ordered in the way specified above.
- The format is: `#{customerName} ::: {flight1}, {flight2}, {flight3}...`

Constraints

- There will be no invalid input lines.
- The `customerName` is a string which may contain any ASCII characters except ' ' (space) and '='.
- The `customerFlight` is an integer in range [0, 10000].

- There will be no non-existent `customerNames` in the commands that require `customerNames`.
- If all data ordering fails, you should order the data by order of input.
- Allowed working time / memory: 100ms / 16MB.

Examples

Input	Output
Donald 1549 4592 3945 111 Prakash 111 45 Gibbs 492 502 Isacc 204 544 I believe I can fly!	#Donald ::: 111, 1549, 3945, 4592 #Gibbs ::: 492, 502 #Isacc ::: 204, 544 #Prakash ::: 45, 111
Prakash 111 134 2451 232 Sony 222 Prakash 555 Stamat 111 Stamat = Sony I believe I can fly!	#Prakash ::: 111, 134, 232, 555, 2451 #Sony ::: 222 #Stamat ::: 222

Solution

```
namespace RainAir
{
    class Program
    {
        static void Main(string[] args)
        {
            var RainAir = new Dictionary<String, List<int>>();
            var line = Console.ReadLine();

            while (line != "I believe I can fly!")
            {
                if (line.IndexOf('=') == -1)
                {
                    var parts = line.Split(' ');
                    string customerName = parts[0];
                    List<int> customerFlights;

                    if (RainAir.ContainsKey(customerName))
                    {
                        customerFlights = RainAir[customerName];
                    }
                    else
                    {
                        customerFlights = new List<int>();
                    }
```

```
                for (int i = 1; i < parts.Count(); i++)
                {
                    customerFlights.Add(int.Parse(parts[i]));
                }

                if (RainAir.ContainsKey(customerName))
                {
                    RainAir[customerName] = customerFlights;
                }
                else
                {
                    RainAir.Add(customerName, customerFlights);
                }
            }
            else
            {
                var parts = line.Replace(" = ", "=").Split('=');
                if (RainAir.ContainsKey(parts[0]))
                {
                    RainAir[parts[0]] = RainAir[parts[1]];
                }
            }
            line = Console.ReadLine();
        }

        foreach (var customer in RainAir.OrderByDescending(c =>
c.Value.Count).ThenBy(c => c.Key))
        {
            Console.Write("#{0} ::: ", customer.Key);
            Console.WriteLine(String.Join(", ", customer.Value.OrderBy(a =>
a)));
        }
    }
}
```

Съдържание