Упражнения: Методи

Тествайте решенията си в **Judge системата**: https://judge.softuni.bg/Contests/3160/Methods

1. Оценки

Напишете метод, който **чете оценка** между **2.00** и **6.00** и **отпечатва съответната оценка с думи:**

- 2.00 – 2.99 - "Fail"
- 3.00 – 3.49 - "Poor"
- 3.50 – 4.49 - "Good"
- 4.50 – 5.49 - "Very good"
- 5.50 – 6.00 - "Excellent"

Примери

Вход	Изход
3.33	Poor
4.50	Very good
2.99	Fail

Насоки

1. Прочетете оценката от конзолата и я подайте на метода **PrintInWords**.

```
using System;

public class Test
{
    public static void Main()
    {
        double grade = double.Parse(Console.ReadLine());

        PrintInWords(grade);
    }
```

2. Създайте метода **PrintInWords** и добавете **условни конструкции** за всяка оценка.

```
private static void PrintInWords(double grade)
{
    if (grade >= 2.00 && grade <= 2.99)
    {
        Console.WriteLine("Fail");
    }

    //TODO: make the rest
}
```

2. Знак на цели числа

Създайте метод, който чете цяло число **n** и отпечатва дали числото е **положително**, **отрицателно** или **равно на 0:**

Примери

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

3. Изчисления

Напишете програма, която на първия ред получава **стринг** ("**add**", "**multiply**", "**subtract**" or "**divide**"), а на следващите **два реда** получава **две цели числа.** Създайте **четири метода** (по един за всяка операция) и **извикайте правилния метод** в зависимост от командата. Методът трябва да **отпечатва резултата от пресмятането**.

Примери

Вход	Изход
subtract 5 4	1
divide 8 4	2

Насоки

1. **Прочетете командата** на първия ред и **двете числа** и след това направете **if/switch конструкция** за всяка отделна операция:

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

    switch (command)
    {
        case "add":
            Add(a, b);
            break;
        case "subtract":
            Subtract(a, b);
            break;

        //TODO: check for the rest of the commands
    }
}
```

2. Създайте **четири метода** (по един за всяка операция) и **отпечатайте** резултата:

```
private static void Multiply(int a, int b)
{
    Console.WriteLine(a * b);
}

private static void Divide(int a, int b)
{
    Console.WriteLine(a / b);
}

//TODO: create the rest of the methods
```

4. Отпечатване на триъгълник

Създайте метод, който **отпечатва триъгълник** с различни размери, както е показано в следните примери:

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

Насоки

1. **Прочетете данните** от входа
2. Започнете, като създадете метод, който **отпечатва един ред** от **зададено начало** до **зададен край**. Изберете **описателно име** за метода, което да **отразява неговата цел**:

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

3. Създайте друг метод, който да **отпечатва целия триъгълник.**
4. Помислете как можете, за да решите задачата.
5. Един от начините да използвате метода **PrintLine**, за да решите задачата, е като създадете **два for цикъла:**
6. Първият може да отпечатва **първата половина** от триъгълника:

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

7. Вторият може да отпечатва **втората половина** от триъгълника.

```
for (int i = n - 1; i >= 1; i--)
{
    PrintLine(1, i);
}
```

5. Лице на правоъгълник

Създайте метод, който изчислява и **връща** лицето на правоъгълник при зададени височина и ширина:

Примери

Вход	Изход
3 4	12
6 2	12

Насоки

1. Прочетете входа
2. Създайте метод, но този път вместо да използвате "**static void**" преди името му, използвайте **"static double",** за да **върнете** стойност от тип **double:**

```
static double GetRectangleArea(double width, double height)
{
    return width * height;
}
```

3. **Извикайте** метода GetRectangleArea в Main и **запазете върнатата стойност в нова променлива:**

```
double width = double.Parse(Console.ReadLine());
double height = double.Parse(Console.ReadLine());
double area = GetRectangleArea(width, height);
Console.WriteLine(area);
```

6. Повторение на стринг

Напишете метод, който **получава стринг** и цяло число **n**, което означава броя повторения. Методът трябва да **върне нов стринг**, който представлява получения текст, повторен **n** пъти.

Примери

Вход	Изход
abc 3	abcabcabc
String 2	StringString

Насоки

1. Прочетете данните от входа
2. Създайте метод (например **RepeatString**) и подайте като аргументи **стринга** и **броя на повторенията:**

```
private static string RepeatString(string str, int count)
{
    string result = "";

    for (int i = 0; i < count; i++)
    {
        //TODO: append the string to the result
    }

    return result;
}
```

3. В метода **Main** отпечатайте резултата.

7. Степени

Напишете метод, който изчислява и връща стойността на дадено число (база), повдигнато на определена степен:

Примери

Вход	Изход
2 8	256
3 4	81

Насоки

1. Прочетете данните от входа
2. Създайте **метод**, който приема **два параметъра** – **числото (базата)** и **степента**, на която трябва да се повдигне. Методът трябва да **връща резултата**, който е от тип **double.**

```
static double RaiseToPower(double number, int power)
{
    double result = 0d;

    // TODO: Calculate result (use a loop, or Math.Pow())

    return result;
}
```

3. **Отпечатайте** резултата

8. По-голяма стойност

Създайте метод **GetMax()**, който **връща по-голямата** от две стойности (стойностите могат да бъдат от тип **int, char** или **string**).

Примери

Вход	Изход
int 2 16	16
char a z	z

string aaa bbb	bbb

9. Произведение от четни и нечетни цифри

Създайте програма, която **умножава сумата** от всички **четни цифри** на дадено число по **сумата** от всички **нечетни цифри** на същото число:

- Създайте метод **GetMultipleOfEvenAndOdds()**
- Създайте метод **GetSumOfEvenDigits()**
- Създайте метод **GetSumOfOddDigits()**
- Може да използвате **Math.Abs()** за отрицателни числа

Примери

Вход	Изход	Обяснение
-12345	54	Четни цифри: 2 и 4 Нечетни цифри: 1, 3 и 5 Сума от четните цифри: 2 + 4 = 6 Сума от нечетните цифри: 1 + 3 + 5 = 9 Произведение на двете суми: 6 * 9 = 54

10. Математически операции

Напишете метод, който получава **две реални числа** и **оператор**, пресмята резултата от дадената операция и го **връща.** Като вход ще получите **три аргумента – първо число**, **оператор** и **второ число.** Възможните оператори са: '**/**', '*****', '**+**', и '**-**'.

Отпечатайте резултата, като го форматирате до **два знака** след **десетичната запетая.**

Примери

Вход	Изход
5 * 5	25
4 + 8	12

Насоки

1. Прочетете данните от входа
2. Създайте метод, който връща стойност от тип **double** (резултата от математическите изчисления)

```
private static double Calculate(int a, string @operator, int b)
{
    double result = 0;

    switch (@operator)
    {
        //TODO: check for all the possible operands and calculate the result
    }

    return result;
}
```

11. Най-малкото от три числа

Напишете метод, който **отпечатва най-малкото** от **три цели числа.**

Примери

Вход	Изход
2 5 3	2
600 342 123	123
25 21 4	4

12. Брой на гласните букви.

Напишете метод, който получава **един стринг** и отпечатва **броя на гласните букви** в него.

Примери

Вход	Изход
SoftUni	3
Cats	1
JS	0

13. Символи в определен диапазон

Напишете метод, който получава **два символа** и отпечатва **на един ред всички символи** между **първия** и **втория**, следвайки **ASCII** таблицата.

Примери

Вход	Изход
a d	b c
# :	$ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9
C #	$ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B

14. Валидатор на пароли

Напишете програма, която **проверява** дали дадена парола е **валидна.** За да бъде валидна, паролата трябва да отговаря на следните **правила:**

- **Дълга** е от **6** до **10 символа** (**включително**)
- Съдържа **само букви** и **цифри**
- Съдържа **най-малко 2 цифри**

Ако паролата е **валидна,** отпечатайте "**`Password is valid`**". Ако **не е валидна**, за всяко правило, което **не е удовлетворено**, отпечатайте съобщение:

- **"Password must be between 6 and 10 characters"**
- **"Password must consist only of letters and digits"**
- **"Password must have at least 2 digits"**

Примери

Вход	Изход
logIn	Password must be between 6 and 10 characters Password must have at least 2 digits
MyPass123	Password is valid
Pass	Password must consist only of letters and digits Password must have at least 2 digits

Насоки

Напишете **отделен метод** за всяко правило.

15. Централен символ

Ще получите **един стринг.** Напишете метод, който отпечатва символът в **средата** на **дадения стринг.** Ако дължината на стринга е **четна**, тогава има **два централни символа.**

Примери

Вход	Изход
aString	r
someText	eT
3245	24

16. Деление на факториел

Напишете програма, която получава **две цели числа**. Изчислете факториела на всяко от числата. **Разделете първия резултат** на **втория** и отпечатайте **частното**, форматирано до **втория знак** след десетичната запетая.

Примери

Вход	Изход
5 2	60.00

Вход	Изход
6 2	360.00

17. Числа палиндроми

Палиндром е число, което се чете по един и същ начин **от ляво надясно** и **от дясно наляво**, например 323 или 1001. Напишете програма, която чете **положителни цели числа**, докато не получи команда **"End"**. За всяко число отпечатайте **дали числото е палиндром**, или **не.**

Примери

Вход	Изход
123 323 421 121 END	false true false true

Вход	Изход
32 2 232 1010 END	false true true false

18. Топ число

Топ число е цяло число, което има следните **свойства:**

- **Сумата** от **цифрите** му **се дели на 8**, например 8, 16, 88
- Съдържа **поне 1 нечетна цифра**, например 232 (съдържа 3), 707 (съдържа 7), 87578 (съдържа 7 и 5)

Напишете програма, която **чете число n** и отпечатва **всички топ числа** в диапазона от **1** до **n.**

Примери

Вход	Изход
50	17 35

Вход	Изход
100	17 35 53 71 79 97

19. * Манипулатор на масиви

Трифон най-сетне е успял да стане junior софтуерен инженер и е получил своята първа задача. Тя е свързана с **манипулиране** на **масив** от **цели числа.** Той не е много доволен от нея, защото не обича работата с масиви.

Но той знае, че заплащането за задачата е щедро, така че е готов да даде на някого да му помогне и да изпълни половината от задачата. От друга страна, вие обичате масивите (и парите), така че решавате да му помогнете със задачата.

Масивът може да бъде манипулиран с една от следните команди:

- **exchange {index} – разделя** масива **след** дадения индекс и **разменя местата** на двата по-малки масива, които са се образували. Пример: [1, 2, 3, 4, 5] -> exchange 2 -> резултат: [4, 5, 1, 2, 3]
  - Ако индексът е **извън границите** на масива, отпечатайте "**Invalid index**"
- **max even/odd** – връща **индекса** на максималния четен/нечетен елемент. Пример: [1, 4, 8, 2, 3] -> max odd -> резултат: 3
- **min even/odd** – връща **индекса** на минималния четен/нечетен елемент. Пример: [1, 4, 8, 2, 3] -> min even -> резултат: 4
  - Ако има **два или повече** минимални/максимални елемента, **върнете индекса** на този, който е **най-вдясно**
  - Ако **няма** минимален/максимален елемент, **отпечатайте "No matches"**
- **first {count} even/odd** – връща първите {count} елемента -> [1, 8, 2, 3] -> **first 2 even** -> резултат [**8, 2]**
- **last {count} even/odd** – връща последните {count} елемента elements -> [1, 8, 2, 3] -> **last 2 odd** -> резултат [**1, 3]**
  - Ако **count** е **по-голям** от дължината на масива, отпечатайте "**Invalid count**"
  - Ако **няма достатъчно елементи**, които да запълнят **count**, отпечатайте **възможно най-много** елементи. Ако **няма нито един четен/нечетен** елемент, отпечатайте **празен масив** "[]"
- **end – спрете** да приемате входни данни и **отпечатайте финалния масив**

Вход

- Входните данни трябва да се прочетат от конзолата
- На първия ред ще получите **началния масив** като поредица от цели числа, **разделени** с **интервал.**
- На следващите редове, до прочитане на команда **“end”,** ще получавате **команди за манипулация** на масива.
- Входните данни винаги ще бъдат валидни и в описания формат. Няма необходимост да ги проверявате експлицитно.

Изход

- Изходът трябва да бъде отпечатан на конзолата:
  - Отпечатвайте резултата от **отделните команди** на **отделни редове**
  - На последния ред отпечатайте **финалния масив** в **квадратни скоби**, като елементите му трябва да са **разделени** със **запетая и интервал** (“, “).

Ограничения

- Броят на входните аргументи ще бъде в диапазона [2 … 50]
- Елементите на масива ще бъдат цели числа в диапазона [0 … 1000]
- Броят на елементите ще бъде в диапазона [1 … 50]
- Индексите ще бъдат в диапазона $[-2^{31} … 2^{31} – 1]$
- Броят на първите/последните елементи ще бъде в диапазона $[1 … 2^{31} – 1]$
- Няма да има излишни интервали във входа.
- Позволено време за изпълнение на програмата: 0.1 секунда. Позволеня памет: 16 MB.

Примери

Вход	Изход
1 3 5 7 9 exchange 1 max odd min even first 2 odd last 2 even exchange 3 end	2 No matches [5, 7] [] [3, 5, 7, 9, 1]
Вход	**Изход**
1 10 100 1000 max even first 5 even exchange 10 min odd exchange 0 max even min even end	3 Invalid count Invalid index 0 2 0 [10, 100, 1000, 1]

Вход	Изход
1 10 100 1000 exchange 3 first 2 odd last 4 odd end	[1] [1] [1, 10, 100, 1000]