Lab: Methods, Debugging and Troubleshooting Code

Problems for exercises and homework for the "Programming Fundamentals" course @ SoftUni.

You can check your solutions here: https://judge.softuni.bg/Contests/304/Methods-and-Debugging-Lab.

I. Declaring and Invoking Methods

1. Blank Receipt

Create a method that prints a blank cash receipt. The method should invoke three other methods: one for printing the header, one for the body and one for the footer of the receipt.

The header should contain the following text:	`CASH RECEIPT` `------------------------------`
The body should contain the following text:	`Charged to_____________________` `Received by_____________________`
And the text for the footer:	`------------------------------` `© SoftUni`

Examples

Output

`CASH RECEIPT`
`------------------------------`
`Charged to_____________________`
`Received by_____________________`
`------------------------------`
`© SoftUni`

Hints

1. First create a method with no parameters for printing the header starting with **static void**. Give it a **meaningful name** like "PrintReceiptHeader" and write the code that it will execute:

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

2. Do the same for printing the receipt body and footer.
3. Create a **method that will call all three methods** in the necessary order. Again, give it a **meaningful and descriptive name** like "PrintReceipt" and write the code:

```
static void PrintReceipt()
{
    PrintReceiptHeader();
    PrintReceiptBody();
    PrintReceiptFooter();
}
```

4. For printing **"©"** use Unicode **"\u00A9"**
5. **Call** (invoke) the PrintReceipt method from the main.

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

2. Sign of Integer Number

Create a method that prints the sign of an integer number n.

Examples

Input	Output
2	The number 2 is positive.
-5	The number -5 is negative.
0	The number 0 is zero.

Hints

1. Create a method with a **descriptive name** like "PrintSign" which should receive **one parameter** of type **int**.

```
static void PrintSign(int number)
{

}
```

2. Implement the body of the method by handling different cases:
    a. If the number is greater than zero
    b. If the number is less than zero
    c. And if the number is equal to zero
3. Call (invoke) the newly created method from the main.

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

3. Printing Triangle

Create a method for printing triangles as shown below:

Examples

Input	Output
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

Hints

1. After you read the input
2. Start by creating a method **for printing a single line** from a **given start** to a **given end**. Choose a **meaningful name** for it, describing its purpose:

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

3. Think how you can use it to solve the problem
4. After you spent some time thinking, you should have come to the conclusion that you will need two loops
5. In the first loop you can print the first half of the triangle without the middle line:

```
for (int i = 0; i < n; i++)
{
    PrintLine(1, i);
}
```

6. Next, print the middle line:

```
PrintLine(1, n);
```

7. Lastly, print the rest of the triangle:

```
for (int i = n - 1; i >= 0 ; i--)
{
    PrintLine(1, i);
}
```

4. Draw a Filled Square

Draw at the console a filled square of size n like in the example:

Examples

Input	Output
4	-------- -\/\/\/- -\/\/\/- --------

Hints

1. Read the input
2. Create a method which will print the top and the bottom rows (they are the same). Don't forget to give it a descriptive name and to give it as a parameter some length
    a. Instead of loop you can use the "new string" command which creates a new string consisting of a character repeated some given times:

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

3. Create the method which will print the middle rows. Well, of course, you should probably name it "PrintMiddleRow"

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

    Console.WriteLine('-');
}
```

4. Use the methods that you've just created to draw a square

```
static void Main()
{
    int n = int.Parse(Console.ReadLine());
    PrintHeaderRow(n);
    // TODO: Draw the rest of the square
}
```

II. Returning Values and Overloading

5. Temperature Conversion

Create a method that converts a temperature from **Fahrenheit** to **Celsius**. Format the result to the 2nd decimal point.

Use the formula: **`(fahrenheit - 32) * 5 / 9`**.

Examples

Input	Output
95	35.00
33.8	1.00
-40	-40.00

Hints

1. Read the input
2. Create a method, which **returns a value of type `double`**:

```
static double FahrenheitToCelsius(double fahrenheit)
{
    return (fahrenheit - 32) * 5 / 9;
}
```

3. **Invoke** the method in the main and **save the return value in a new variable**:

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

6. Calculate Triangle Area

Create a method that calculates and **returns** the area of a triangle by given base and height:

Examples

Input	Output
3 4	6

Hints

1. After reading the input
2. Create a method, but this time **instead** of typing **"static void"** before its name, type **"static double"** as this will make it to **return a value of type double**:

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

3. **Invoke** the method in the main and **save the return value in a new variable**:

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

7. Math Power

Create a method that calculates and returns the value of a number raised to a given power:

Examples

Input	Output
2 8	256
3 4	81

Hints

1. As usual, read the input
2. Create a method which will have two parameters - the number and the power, and will return a result of type double:

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

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

    return result;
}
```

3. Print the result

8. Greater of Two Values

You are given two values of the same type as input. The values can be of type int, char of string. Create a method GetMax() that returns the greater of the two values:

Examples

Input	Output
int 2 16	16
char a z	z
string Ivan Todor	Todor

Hints

1. For this method you need to create three methods with the same name and different signatures
2. Create a method which will compare integers:

```
static int GetMax(int first, int second)
{
    if (first >= second)
    {
        // TODO: return value
    }

    // TODO: handle other cases
}
```

3. Create a second method with the same name which will compare characters. Follow the logic of the previous method:

```
static char GetMax(char first, char second)
{
    // TODO: create logic
}
```

4. Lastly you need to create a method to compare strings. This is a bit different as strings don't allow to be compared with the operators > and <

```
static string GetMax(string first, string second)
{
    if (first.CompareTo(second) >= 0)
    {
        // TODO: return value
    }
    // TODO: return value
}
```

You need to use the method "CompareTo()", which returns an integer value (greater than zero if the compared object is greater, less than zero if the compared object is lesser and zero if the two objects are equal.

5. The last step is to read the input, use appropriate variables and call the GetMax() from your Main():

```
var type = Console.ReadLine();
if (type == "int")
{
    int first = int.Parse(Console.ReadLine());
    int second = int.Parse(Console.ReadLine());
    int max = GetMax(first, second);
    Console.WriteLine(max);
}
else if (type == "char")
{
    // TODO: call GetMax with char arguments
}
else if (type == "string")
{
    // TODO: call GetMax with string arguments
}
```

III. Debugging and Program Flow

9. Multiply Evens by Odds

Create a program that reads an **integer number** and **multiplies the sum of all its even digits** by **the sum of all its odd digits**:

Examples

Input	Output	Comments
12345	54	12345 has **2 even digits** - 2 and 4. Even digits has **sum of 6.** Also it has **3 odd digits** - 1, 3 and 5. Odd digits has **sum of 9.** **Multiply 6 by 9** and you get **54.**
-12345	54	

Hints

1. Create a method with a **name describing its purpose** (like GetMultipleOfEvensAndOdds). The method should have a **single integer parameter** and an **integer return value**. Also the method will call two other methods:

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

2. Create two other methods each of which will sum either even or odd digits
3. Implement the logic for summing odd digits:

```
private static int GetSumOfOddDigits(int n)
{
    int sum = 0;
    while (n > 0)
    {
        int lastDigit = n % 10;
        if (lastDigit % 2 != 0)
        {
            // TODO: add to sum
        }

        n /= 10;
    }

    return sum;
}
```

4. Do the same for the method that will sum even digits
5. As you test your solution you may notice that it doesn't work for negative numbers. Following the program execution line by line, find and fix the bug (**hint: you can use Math.Abs()**)

10. Debug the Code: Holidays Between Two Dates

You are assigned to **find and fix the bugs** in an existing piece of code, using the Visual Studio **debugger**. You should trace the program execution to find the lines of code that produce incorrect or unexpected results.

You are given a program (existing **source code**) that aims to **count the non-working days between two dates** given in format **`day.month.year`** (e.g. between **1.05.2015** and **15.05.2015** there are **5** non-working days – Saturday and Sunday).

Examples

Input	Output	Comments
1.05.2016 15.05.2016	5	There are 5 non-working days (Saturday / Sunday) in this period: 1-May-2016, 7-May-2016, 8-May-2016, 14-May-2016, 15-May-2016
1.5.2016 2.5.2016	1	Only 1 non-working day in the specified period: 1.05.2016 (Sunday)
15.5.2020 10.5.2020	0	The second date is before the first. No dates in the range.
22.2.2020 1.3.2020	4	Two Saturdays and Sundays: • 22.02.2020 and 23.02.2020 • 29.02.2020 and 1.03.2020

You can **find the broken code** in the judge system: Broken Code for Refactoring. It looks as follows:

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);
    }
}
```

Hints

There are **4 mistakes** in the code. You've got to **use the debugger** to find them and fix them. After you do that, submit your **fixed code in the judge contest**: https://judge.softuni.bg/Contests/Practice/Index/304#8.

11. Price Change Alert

You are assigned to **rework a given piece of code** which is working **without bugs** but is **not properly formatted**.

The given program **tracks stock prices** and **gives updates** about the **significance in each price change**. Based on the significance, there are **four kind of changes**: no change at all (price is equal to the previous), minor (difference is below the significance threshold), price up and price down.

You can **find the broken code** here: Broken Code for Refactoring.

Input

- On the first line you are given **N** - the number of prices
- On the second line you are given the significance threshold
- On the next N lines, you are given prices

Output

- Don't print anything for the first price
- If there is **no difference** from the previous price the output message is: "NO CHANGE: {current price}"
- In case of **minor change**: "MINOR CHANGE: {last price} to {current price} ({difference}%)"
- In case of **major change**: "PRICE UP: {last price} to {current price} ({difference}%)" or "PRICE DOWN: {last price} to {current price} ({difference}%)"

The percentage should be rounded to the second digit after the decimal point.

Examples

Input	Output

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%)

Hints

1. Download the source code and get familiar with it
2. Deal with poor code formatting - Remove unnecessary blank lines, indent the code properly
3. Fix method parameters naming
4. Give methods a proper name