# Exam Preparation

Test your tasks in the Judge system: https://judge.softuni.org/Contests/4457/Exam-Preparation-III

## 1. English Name of Each Digit

Write a program that:

- Reads an **integer positive number** from the console
- Print **English name** of the each digit of a given number on a separate line, **starting from the last to the first digit**
  - 1 -> "one"
  - 2 -> "two"
  - 3 -> "three"
  - 4 -> "four"
  - 5 -> "five"
  - 6 -> "six"
  - 7 -> "seven"
  - 8 -> "eight"
  - 9 -> "nine"

### Example

| Input | Output |
|---|---|
| 512 | two<br>one<br>five |
| 121 | one<br>two<br>one |
| 1643 | three<br>four<br>six<br>one |

## 2. Average Last Elements

Write a program that:

- Reads an **array of integer numbers** from the first line of the console, separated by single space
- Read an **integer number N** from the second line of the console
- Find **average value** of the **last N elements** in the array
- Print the **average value** formatted to the **second decimal digit**

## Example

| Input | Output | Comments |
|---|---|---|
| 3 42 61 7 8 9 10 23<br>4 | 12.50 | Last 4 numbers in the array are: 8 9 10 23<br>Average value is: (8 + 9 + 10 + 23) / 4 = 12.5 |
| 12 34 98 42 65 12<br>3 | 39.67 | Last 3 numbers in the array are: 42 65 12<br>Average value is: (42 + 65 + 12) / 3 = 39.67 |

# 3. Unit Test Method: Extract File

Test a given method which takes in **a path in the form of a string** and gets the **filename** and **file extension** (if it exists).

## Examples

| Input | Output |
|---|---|
| C:\Users\John\Documents\example.txt | File name: example\nFile extension: txt |
| C:\Users\John\Documents\example | File name: example |
| C:\Users\John\Documents?\example.txt | File name: example\nFile extension: txt |
| null | Throws ArgumentNullException |
| (empty string) | Throws ArgumentNullException |

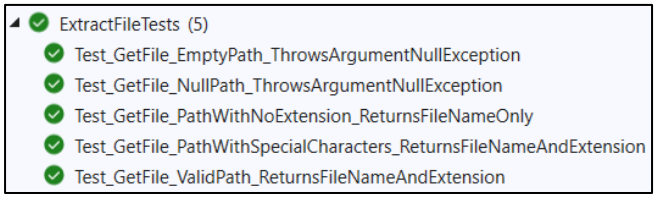
The method is found in the **`ExtractFile.cs`** file:

```
public class ExtractFile
{
    5 references
    public static string GetFile(string? path)
    {
        if (string.IsNullOrEmpty(path))
        {
            throw new ArgumentNullException(nameof(path));
        }

        string[] pathParts = path.Split(separator: @"\");
        string file = pathParts[^1];

        if (file.IndexOf('.') == -1)
        {
            return $"File name: {file}";
        }

        return $"File name: {file.Split(separator: '.')[0]}\nFile extension: {file.Split(separator: '.')[1]}";
    }
}
```

You are given a **test file `ExtractFileTests.cs`** containing **5 empty tests**. Implement all the unit tests:

```
public class ExtractFileTests
{
    [Test]
    0 references
    public void Test_GetFile_NullPath_ThrowsArgumentNullException()...

    [Test]
    0 references
    public void Test_GetFile_EmptyPath_ThrowsArgumentNullException()...

    [Test]
    0 references
    public void Test_GetFile_ValidPath_ReturnsFileNameAndExtension()...

    [Test]
    0 references
    public void Test_GetFile_PathWithNoExtension_ReturnsFileNameOnly()...

    [Test]
    0 references
    public void Test_GetFile_PathWithSpecialCharacters_ReturnsFileNameAndExtension()...
}
```

When you are ready make sure your **tests run:**




**IMPORTANT: DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.**

## 4. Unit Test List: Drum Set

Test a given method which takes in **a decimal representing money, list of integers representing quality of drums and a list of strings representing commands**.

For each command **each drum is hit by the number indicated**. If a drum breaks (**goes below or equal to zero**) it must be replaced by the **initial amount,** then **initial amount times three** must be reduced from the money. If a new drum can't be afforded it is removed from the list.

When the command **"Hit it again, Gabsy!"** is received, the method ends, and returns the current **state** of the **drum set** alongside the **leftover money**.

The method is found in the **`DrumSet.cs`** file:

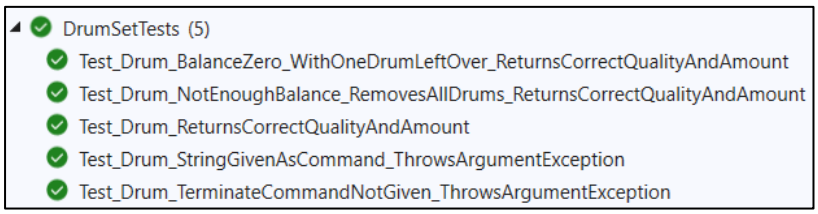
```
public class DrumSet
{
    5 references
    public static string Drum(decimal money, List<int> initialQuality, List<string> commands)
    {
        List<int> usedQuality = initialQuality.ToList();

        foreach (string command in commands)
        {
            if (command == "Hit it again, Gabsy!")
            {
                return $"{string.Join(" ", usedQuality)}\nGabsy has {money:f2}lv.";
            }

            bool isParsed = int.TryParse(command, out int power);
            if (!isParsed)
            {
                throw new ArgumentException(message: "Number did not parse correctly!");
            }
```

```
            for (int i = 0; i < usedQuality.Count; i++)
            {
                usedQuality[i] -= power;
                if (usedQuality[i] > 0)
                {
                    continue;
                }

                int price = initialQuality[i] * 3;
                if (money - price > 0)
                {
                    money -= price;
                    usedQuality[i] = initialQuality[i];
                }
                else if (money - price <= 0)
                {
                    initialQuality.RemoveAt(i);
                    usedQuality.RemoveAt(i);
                    i--;
                }
            }
        }

        throw new ArgumentException(message: "Terminate command not given!");
    }
}
```

You are given a **test file `DrumSetTests.cs`** containing **5 empty tests**. Implement all the unit tests:

```
public class DrumSetTests
{
    [Test]
    0 references
    public void Test_Drum_TerminateCommandNotGiven_ThrowsArgumentException()...

    [Test]
    0 references
    public void Test_Drum_StringGivenAsCommand_ThrowsArgumentException()...

    [Test]
    0 references
    public void Test_Drum_ReturnsCorrectQualityAndAmount()...

    [Test]
    0 references
    public void Test_Drum_BalanceZero_WithOneDrumLeftOver_ReturnsCorrectQualityAndAmount()...

    [Test]
    0 references
    public void Test_Drum_NotEnoughBalance_RemovesAllDrums_ReturnsCorrectQualityAndAmount()...
}
```

When you are ready make sure your **tests run:**




**IMPORTANT: DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.**