# Exercises: Recursion

You can check your solutions here: https://judge.softuni.bg/Contests/3185/Recursion.

## 1. Recursive Array Sum

Write a program that finds the sum of all elements in an integer array. Use **recursion**.

**Note**: In practice recursion should not be used here (instead use an **iterative solution**), this is just an exercise.

### Examples

| Input | Output |
|---------|--------|
| 1 2 3 4 | 10 |
| -1 0 1 | 0 |

### Hints

Write a **recursive** method. It will take as arguments the **input array** and the **current index**.

- The method should return the **current element** + the **sum of all next elements** (obtained by recursively calling it).
- The recursion should stop when there are no more elements in the array.

```csharp
private static int Sum(int[] array, int index)
{
    // TODO: Set bottom of recursion
    // TODO: Return the sum of current element + sum of elements to the right
}
```

## 2. Reverse Array

Write a program that **prints given array in reversed order**, using **recursion**.

### Examples

| Input | Output |
|-------------|-------------|
| 1 2 3 4 5 6 | 6 5 4 3 2 1 |

## 3. Recursive Factorial

Write a program that finds the factorial of a given number. Use **recursion**.

**Note**: In practice recursion should not be used here (instead use an **iterative solution**), this is just an exercise.

### Examples

| Input | Output |
|-------|---------|
| 5 | 120 |
| 10 | 3628800 |

### Hints

Write a **recursive** method. It will take as arguments an integer number **n**.

---

- The method should return the **current element** * the **result of calculating factorial of current element - 1** (obtained by recursively calling it).
- The recursion should stop when there are no more elements in the array.

```
static long Factorial(int n)
{
    // TODO: Set bottom of recursion
    // TODO: Return the multiple of current n and factorial of n - 1
}
```

- Note: the above will be **slow** when the number **n** is big enough. You can speed-up the calculation by remembering in array each value, which is already calculated.

# 4. Recursive Drawing

Write a program that draws the figure below depending on n. Use **recursion**.

## Examples

| Input | Output |
|-------|--------|
| 2 | `**`<br>`*`<br>`#`<br>`##` |
| 5 | `*****`<br>`****`<br>`***`<br>`**`<br>`*`<br>`#`<br>`##`<br>`###`<br>`####`<br>`#####` |

## Hints

Set the bottom of the recursion.

```
if (n <= 0)
{
    return;
}
```

Define pre- and post- recursive behavior

```
Console.WriteLine(new string('*', n));
PrintFigure(       );
Console.WriteLine(new string('#', n));
```

# 5. * Nested Loops to Recursion

Write a program that **simulates the execution of n nested loops from 1 to n** which prints the values of all its iteration variables at any given time on a single line. **Use recursion**.

## Examples

| Input | Output | Solution with nested loops (assuming n is positive) |
|-------|--------|----------------------------------------------------|
| 2 | 1 1<br>1 2<br>2 1<br>2 2 | ```csharp\nint n = 2;\n\nfor (int i1 = 1; i1 <= n; i1++)\n{\n    for (int i2 = 1; i2 <= n; i2++)\n    {\n        Console.WriteLine($"{i1} {i2}");\n    }\n}\n``` |
| 3 | 1 1 1<br>1 1 2<br>1 1 3<br>1 2 1<br>1 2 2<br>…<br>3 2 3<br>3 3 1<br>3 3 2<br>3 3 3 | ```csharp\nint n = 3;\n\nfor (int i1 = 1; i1 <= n; i1++)\n{\n    for (int i2 = 1; i2 <= n; i2++)\n    {\n        for (int i3 = 1; i3 <= n; i3++)\n        {\n            Console.WriteLine($"{i1} {i2} {i3}");\n        }\n    }\n}\n``` |

## Hints

- Read about **recursive nested loops** here: https://introprogramming.info/english-intro-csharp-book/read-online/chapter-10-recursion.