# Exercises: Basic Algorithms

Problems for exercises and homework for the ["Java Advanced" course @ SoftUni](https://softuni.bg/trainings/2246/java-advanced-january-2019).

You can check your solutions here: <https://judge.softuni.bg/Contests/1561>.

## 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.

## 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.

* 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.

## Sum of Coins

Let’s have a **range of possible coin values** (e.g. 1, 2, 5, 10, 20, 50) and a **desired sum**. In this problem, the goal is to **reach the sum using as few coins as possible using a greedy approach**. We’ll assume that each coin value and the desired sum are **integers**, but we can easily modify the problem to accommodate floating-point values.

**Examples:**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| Coins: 1, 2, 5, 10, 20, 50  Sum: 923 | Number of coins to take: 21  18 coin(s) with value 50  1 coin(s) with value 20  1 coin(s) with value 2  1 coin(s) with value 1 | 18\*50 + 1\*20 + 1\*2 + 1\*1 = 900 + 20 + 2 + 1 = 923 |
| Coins: 1  Sum: 42 | Number of coins to take: 42  42 coin(s) with value 1 |  |
| Coins: 3, 7  Sum: 11 | Error | Cannot reach desired sum with these coin values |
| Coins: 1, 2, 5  Sum: 2031154123 | Number of coins to take: 406230826  406230824 coin(s) with value 5  1 coin(s) with value 2  1 coin(s) with value 1 | Solution should be fast enough to handle a combination of small coin values and a large desired sum |
| Coins: 1, 9, 10  Sum: 27 | Number of coins to take: 9  2 coin(s) with value 10  7 coin(s) with value 1 | Greedy approach produces non-optimal solution (9 coins to take instead of 3 with value 9) |

### Greedy Approach to Solution

For this problem, a greedy algorithm will attempt to take the best possible coin value (which is the largest), then take the next largest coin value and so on, until the sum is reached or there are no coin values left.

There may be a different amount of coins to take for each value. In one of the examples above we had a very large desired sum and relatively small coin values, which means we’ll need to take a lot of coins. It would not be efficient (and may even cause an Exception) if we return the result as a List<Integer>; a more practical way to do it is to use a **Map<Integer, Integer>** where the keys are the coin values and the values are the number of coins to take for the specified coin value. Therefore, in the second example (coin values = { 1 }, sum = 42), instead of returning a list with 42 elements in it, we’ll return a dictionary with a single key-value pair: 1 => 42.

## Sorting

Implement any or many of the following sorting algorithms:

* Insertions Sort
* Bubble Sort
* Shell Sort
* Merge Sort
* Quicksort
* Bucket Sort

Read a sequence of numbers from the console, sort them and print them back.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 4 3 2 1 | 1 2 3 4 5 |
| 1 4 2 -1 0 | -1 0 1 2 4 |

Here is a breakdown of judge tests:

|  |  |  |
| --- | --- | --- |
| **Test N** | **N of Elements** | **Comments** |
| 001 | 1k | Random |
| 002 | Sorted |
| 003 | Reversed |
| 004 | Duplicate |
| 005 | 10k | Random |
| 006 | Sorted |
| 007 | Reversed |
| 008 | Duplicate |
| 009 | 100k | Random |
| 010 | Sorted |
| 011 | Reversed |
| 012 | Duplicate |

## Searching

Implement any or many of the following sorting algorithms:

* Linear Search
* Binary Search
* Fibonacci Search

Read a sequence of numbers on the first line and a single number on the second from the console. Find the index of the number in the given array. return -1 if the element is not present in the array.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 2 3 4 5  1 | 0 |
| 1 2 3 4 5  6 | -1 |