# C++ Advanced – Exam 2 (2 Dec 2018)

Write C++ code for solving the tasks on the following pages.

Code should compile under the C++03 or the C++11 standard.

Submit your solutions here: https://judge.softuni.bg/Contests/1334/CPlusPlus-Advanced-Exam-2-2-Dec-2018

Any code files that are part of the task are provided under the folder **Skeleton**.

Please follow the exact instructions on uploading the solutions for each task.

# Task 4 – Nodes

You are given parts of code from a program that does computation using a network of Nodes. A Node is a step in the computation, which can connect and send/receive data from other nodes. Different types of nodes do different types of computations on the data they receive:

* An int node stores the data it gets as characters, which are digits in a number ('0'-'9') and converts it into a non-negative integer number **result**. The node expects the sequence to end in a '.' (dot)
* A reverse node stores data it receives and reverses its order when a '.' (dot) is encountered (the dot is not included in the result), and treats that as its **result**
* A sum node stores its data as integer numbers and sums them. It produces the sum as its **result** when it receives a value of 0
* A filter node ignores any data matching certain **predefined** values, and its **result** is each data item that is not one of those predefined values
* An input node is used for reading all the input data for the **computation** from a stream. It treats each input item it reads as a **result**. Only 1 such node will exist for one execution of the program
* An output node prints all the data it receives, separated by single spaces on the console. It can print the data directly, or it can print it as integer numbers. Multiple such nodes can exist. The output node has no **result**

When each node produces its **result**, it **sends** it to all the **nodes it is connected to**. Sending a result to a Node is done by calling the method process on the receiving Node.

The program’s input is separated into 3 parts – the **node definitions**, the **connections** and the **computational** **input**. All 3 parts are read by the provided code, and handled by a NodeNetwork class – it handles **storing the nodes**, handles commands for **connecting one node to another** (meaning the first node sends its results to the second node), and handles the compute operation (read all input through the single input node and process it).

Your task is to study the provided code, any classes and OOP interfaces in it, and implement the missing Node definitions.

### Restrictions

There will be no invalid ids, no invalid connections (i.e. connections which would produce invalid output considering the definitions above) and no loops in the network. No two nodes will have the same id.

The input will always end in '|' – the provided InputNode handles reading that input and stops reading when that symbol is reached.

### Helpers

In the Helpers folder, you will find classes that implement some of the common operations you will need - you may add them to your solution if you decide to use them (they are not part of the Skeleton, so you have to submit them with your files if you want to use them).

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 input  2 reverse  3 filter abc  4 int  5 sum  6 output number  7 output  end  1 2  2 7  1 3  3 4  4 5  5 6  end  1a.2b.3c.0.| | a 1 b 2 c 3 0 6 |