# C++ Advanced – Exam Retake (15 Mar 2020)

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

Code should compile under the C++11 standard.

Submit your solutions here: https://judge.softuni.bg/Contests/1813/CPlusPlus-Advanced-Exam-15-Mar-2020

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 2 – Memory Monitor

Your task is to write a program that allocated/deallocates dynamic memory while it monitors the total occupation of dynamic memory for the application in **bytes**.

**An implementation** for the **MemoryMonitor** class must be provided.

Different commands will be read from the console (the first row from input correspond to how many commands).

```
enum InputCommands
{
	PUSH_NODE                = 0,
	POP_NODE                 = 1,
	PRINT_MEMORY_OCCUPATION  = 2
};
```

- On `PUSH_NODE` command - a dynamic memory with requested size must be **allocated** and the data for this allocation must be kept within the **MemoryMonitor** class (in the **_nodes** member).
  As a result the function should **print** to the console "`Pushed node with memory occupation: `" followed by how many bytes of data were dynamically allocated. End the line with a **newline**.

- On `POP_NODE` command – the last requested dynamic memory node (from the **_nodes** member) must be popped and its memory **deallocated**.
  As a result the function should **print** to the console "`Popped node with memory occupation: `" followed by how many bytes of data were dynamically allocated.
  If there are no nodes to be popped a message "`No nodes to pop`".
  In both cases end the line with a **newline**.

- On `PRINT_MEMORY_OCCUPATION` command – **print** the number of bytes that are **dynamically allocated** for the first **N** MemoryNodes from the _nodes member.
  Use the syntax "`Memory occupation for first N nodes is: SIZE`". End the line with a **newline**.
  Keep in mind that `PRINT_MEMORY_OCCUPATION` can be invoked with random argument **N** that may be bigger or smaller than the currently present number of MemoryNode's inside your **_nodes** struct.
  If **N** happens to be smaller – print the memory occupation only for that number of MemoryNode's.
  If **N** happens to be bigger – print the number N in the message but display memory occupation only for your application existing MemoryNodes.

## Restrictions

You should only submit **.h** and **.cpp** files compressed in a **.zip** archive.
There should be no folders in your **.zip** archive.

Keep in mind that Judge is running on a 64-bit Windows platform where **sizeof(int)** yields **4 bytes.**

## Examples

| Input | Output |
|---|---|
| 4<br>0 5<br>2 1<br>0 3<br>2 1 | Pushed node with memory occupation: 20<br>Memory occupation for first 1 memory nodes is: 20<br>Pushed node with memory occupation: 12<br>Memory occupation for first 1 memory nodes is: 20 |
| 4<br>1<br>1<br>0 30000<br>2 10 | No nodes to pop<br>No nodes to pop<br>Pushed node with memory occupation: 120000<br>Memory occupation for first 10 memory nodes is: 120000 |
| 10<br>0 25<br>1<br>2 1<br>0 20<br>0 5<br>2 2<br>1<br>2 2<br>1<br>1 | Pushed node with memory occupation: 100<br>Popped node with memory occupation: 100<br>Memory occupation for first 1 memory nodes is: 0<br>Pushed node with memory occupation: 80<br>Pushed node with memory occupation: 20<br>Memory occupation for first 2 memory nodes is: 100<br>Popped node with memory occupation: 20<br>Memory occupation for first 2 memory nodes is: 80<br>Popped node with memory occupation: 80<br>No nodes to pop |