C++ Advanced – Exam 2 (21 Apr 2019)

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/1608/CPlusPlus-Advanced-Retake-21-Apr-2019

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 Allocator Reforged

Your task is to write a memory allocator, which is not introducing a memory leak. You are given the `main()` function, which reads a single value (as integer number) of memory followed by N command lines.

- The integer value indicates the number of following command lines (N) you need to process and execute (in the range `[0, INT_MAX]` inclusive).
- The next **N** lines of indicate the command that you should process and execute.
  The commands have the following syntax:
  - "Allocate Single INDEX" – allocate memory for single integer on INDEX in your memory allocator;
  - "Allocate Multiple INDEX" – allocate memory for ALLOCATION_MULTIPLE_SIZE (see Defines.h) on INDEX in your memory allocator;
  - "Deallocate Single INDEX" – deallocate memory for single integer on INDEX in your memory allocator;
  - "Deallocate Multiple INDEX" – deallocate memory for multiple integers on INDEX in your memory allocator;
    Where INDEX can be any integer in the range (in the range `[0, 9]` inclusive);

You should implement the functions **executeCommand()** and **printResult()** in another .cpp file. (For example MemoryAllocatorReforged.cpp)

For each executed command in the **executeCommand()** – you should print a status message depending on the received **ErrorCode** in **printResult()**. Every call to **printResult()** should end with a **newline**.

You should print:

- For successful allocation/deallocation (not introducing memory leak or crashing the problem) – "**command** - success"
- For preventing a memory leak – "**command** - memory leak prevented, will not make allocation"
- For preventing a system crash – "**command** - system crash prevented, will skip this deallocation"
- For trying to deallocate a 'Single' memory node with deallocate 'Multiple' command and vice versa – "**command** – Warning, allocate/deallocate mismatch, will skip this deallocation"
  Where "**command**" is the exact same string that is passed to the function.

Your task is to study the code and implement the function so that the code accomplishes the task described.

You should submit a single `.zip` file for this task, containing **ONLY** the files you created.

The Judge system has a copy of the other files and will compile them, along with your file, in the same directory.

Restrictions

You are free to implement another function/functions that are used internally by the **executeCommand()** and **printResult().**

Each command buffer INDEX will always be in range **[0-9]** inclusive.

Your program is NOT allowed to leak memory on an Allocate/Deallocate action on the same INDEX.

Examples

Input	Output

`4`
`Allocate Single 3`
`Deallocate Single 3`
`Allocate Multiple 1`
`Deallocate Multiple 1`
	`Allocate Single 3 - success`
	`Deallocate Single 3 - success`
	`Allocate Multiple 1 - success`
	`Deallocate Multiple 1 - success`

`4`
`Allocate Single 1`
`Deallocate Multiple 3`
`Allocate Multiple 2`
`Deallocate Multiple 1`
	`Allocate Single 1 - success`
	`Deallocate Multiple 3 - system crash prevented, will skip this deallocation`
	`Allocate Multiple 2 - success`
	`Deallocate Multiple 1 - Warning allocate/deallocate mismatch, will skip this deallocation`

`5`
`Allocate Single 1`
`Allocate Multiple 1`
`Deallocate Single 1`
`Allocate Multiple 1`
`Deallocate Multiple 1`
	`Allocate Single 1 - success`
	`Allocate Multiple 1 - memory leak prevented, will not make allocation`
	`Deallocate Single 1 - success`
	`Allocate Multiple 1 - success`
	`Deallocate Multiple 1 - success`