Programming in C/C++

Exercises 5
!!! Deadline: Tue, Dec 3rd, 23:55 !!!

Task 01:
Implement a class called **`Stack`** capable of manage a stack of `int`s. The stack should be stored on the heap as an array (not a linked list!) and should implement the following specification:

- The constructor shall have a parameter, defining the maximum capacity of the stack with a default value of 10. It shall allocate the array and print the message:
  `"Ctor: I am allocating space for <size> integers"`
- Add destructors if needed. If you add a destructor, it should print the following:
  `"Dtor: I am deallocating a stack of <size> integers filled with <count> values."`
- The class shall also have a copy constructor, copy assignment operator, move constructor, and move assignment operator. For each of them print respectively the following messages
  `"Cctor called", "Cop= called", "Mctor called", "Mop= called"`
  Hint: If you need to copy something, look at memcpy(). Also not that the moved-from object should still exist in a valid (even though undefined state). So make sure to set the stack length & capacity of the old object to 0 after moving.

- The public method "`bool is_full()`" shall return `true` if the stack is full, `false` otherwise.
- The public method "`bool is_empty()`" shall return `true` if the stack is empty, `false` otherwise.
- The public method "`void push(int)`" shall push the given parameter on the stack if the stack is not full, otherwise it shall print the following message:
  `"The stack is full"`
- The public method "`int pop()`" shall return the top value of the stack if the stack is not empty, otherwise it shall return 0 and print the following message:
  `"The stack is empty"`
- The public method "`void show() const`" should print out each stored parameter of the stack in a single line, with single spaces between numbers.
- The public method "`int capacity() const`" shall return the capacity of the Stack, while the public method "`int length() const`" shall return the number of elements currently in the Stack
- An assignment of a single integer to a stack variable shall not be possible.

Provide a header file **stack.hpp** with the class definition and a source file **stack.cpp** with its implementation. Test your implementation with provided **stacksimpletest.cpp**.

Submission: Include the test file with your submission and use it in your Makefile. The final program should be called **stack**.
Points: (code 60 pts, comments 20 pts)

Task 02:
Consider the following program

```
#include <iostream>
int main(void) {
      std::cout << "Hello world!" << std::endl;
      return 0;
}
```

Modify it to produce the output

```
---start---
Hello world!
---end---
```

without changing the main() function and without preprocessor operations. Use C++ features of Chapter 4. Add a report describing why your solution works.

Submission: Submit your code and a Makefile for the executable called **`helloext`**
Points: (code 10 pts, report 10 pts)

Task 03 - **Bonus task**:
Use your StopWatch implementation of exercise 4 to create a preprocessor macro in order to measure a set of instructions. Split your stopwatch implementation of exercise 4 into a header and a source code file. The macro should be called `MEASURETIME`, with the signature:

```
MEASURETIME(description, repetitions, instructions2test)
```

and should be able to compute the time that requires to perform all the code instructions contained in the "`instructions2test`" for "`repetitions`" number of times. The macro should produce the following output at the end of execution:

"My computer requires <t> seconds to execute <description> <repetitions> times"

Create a test program that includes both the `Stack` and the `StopWatch` headers, create a Stack of size 100000 and fill it with random elements as base operation. Then add the following variations to the base operation and test each with 10000 repetitions using your macro above (and the descriptions specified):

- Add code that calls the copy assignment operator of `Stack` with macro description "Copy assignment test"
- Add code that calls the move assignment operator of `Stack` (do not use `std::move`) with macro description "Move assignment test".
- You may remove the print output lines in all the constructors of `Stack` before submitting your solution for this task so as to reduce the amount of output. (*Hint*: they might be helpful during your testing to verify that the right constructors are being called)

Write a report that includes your measurements when executing the test program and explain the root cause for the differences. Note: if your computer is fast, you might have to increase the number of repetitions to see a difference.

Submission: Include the code for the StopWatch and the Stack (both headers and implementation files), as well as the test program with the macro in your submission. Your executable should be called **timetest**

Points: (code 15 pts, report 5 pts)