UNIVERSITÄT DUISBURG ESSEN

NES Networked Embedded Systems

Programming in C/C++

Exercises 5

1. Implement a stack of ints, which is stored on the heap as array (not a linked list!).
   - The constructor shall have a parameter, defining the maximum size of the stack with a default of 10. It shall allocate the array.
   - Add destructors if needed.
   - The class shall also have a copy constructor, copy assignment operator, move constructor, and move assignment operator. Hint: If you need to copy something, look at memcpy().
   - All constructors, destructors and assignment operators shall print out a short message when they are called.
   - The method “bool isfull()” shall return true if the stack is full, false otherwise.
   - The method “bool isempty()” shall return true if the stack is empty, false otherwise.
   - The method “void push(int)” shall push the given parameter on the stack if the stack is not full, otherwise it shall print a message.
   - The method “int pop()” shall return the top value of the stack if the stack is not empty, otherwise it shall print a message and return 0.
   - An assignment of a single integer to a stack variable shall not be possible.

Provide a header file with the class definition and a source file with its implementation. Test your implementation with provided stacktest.cpp.
Note: You might need to compile with “-std=c++11” switch to enable rvalue references. (code 60 pts, comments 20 pts)

2. Consider the following program:

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

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 header comment/report describing why your solution is working. (code 10 pts, report 10 pts)

3. Bonus task: Enclose the output of messages in the stack class’ constructors, destructors, assignment operators with a preprocessor statement to enable conditional debugging. Split your stopwatch implementation of exercise 4 into a header and a source code file. Create a test program that includes both the stack and the stopwatch. Add code that calls the copy assignment operator of Stack and code that calls the move assignment operator of Stack. The statements of these code fragments should be as similar as possible (mostly except a function call). Surround these fragments with a loop (you might need some 1000 iterations) and measure their execution time. Do you see a difference? (code 10 pts, comments 5 pts, report 5 pts)