COMP241
Fall 2015
Programming Assignment 2
Assigned: 8 October, 2015
Deadline: 15 October, 2015

Submission guidelines:

- Make sure you start early -- it might be harder than you think!
- No early deadline this time, but there is an extra credit option
- Please post your solution on Moodle. It is due ***at the beginning of class.***
- Use MS Visual Studio for the assignment. Submit your source code files (.cpp and .h files) separately for each problem.
- Include a comment with the course number, term, assignment, your name, and the date, at the top of each file.
- In addition to writing excellent code, you will be graded on the user experience you create. Any time there are user prompts and input/output, make sure your code is both informative and polite.
- Although I encourage collaboration if you want to work with a classmate, you must turn in your own work. Duplicate code and plagiarism will be considered a violation of the honor code and handled accordingly.

Problem 1
30 points

Implement an ADT class for a stack with an underlying array. Like the Unsorted List from the first Programming Assignment, your class need not be generic. Instead, create a class to support a stack of integers.

You must implement all of the basic functions for a stack, including push, pop, and top/peek. Assume the maximum size of the stack is 10.

You must also implement 3 constructors for your stack:

- Default constructor: Takes no parameters and makes an empty stack
- Item constructor: Takes one item and creates a new stack with the given item as its only member
- Copy constructor: Makes a copy of an existing stack

Submit both your .h and .cpp file for your Stack class.

Problem 2
25 points
Now, you're going to test out your Stack class by writing a driver that calls several test cases demonstrating the behavior of your stack (essentially, a manual test suite).

The nature of the driver is up to you, but you must make sure all of the following test cases are included:

- Create an empty stack and push items on to it, pop items off of it
- Create a single-item stack and remove the item
- Create a stack with several items and make a copy of it
- Print out a stack's size
- Attempt to push an item on to a full stack
- Attempt to pop an item off an empty stack
- Attempt to peek at an item of an empty stack
- Attempt to create a stack with the wrong data type

Remember that user experience is important in your grade. When I run your code, I don't need to necessarily input any information, but I should understand clearly (1) what test you are running, and (2) whether you succeeded or failed.

Problem 3
45 points
For this problem, write a C++ program to determine whether a palindrome contains a string. This is a modification from the book's palindrome example on pages 306-307; use that as a starting point.

Your algorithm should work as follows:

- Prompt the user for an input string (you may assume the user enters only alpha characters; don't worry about error checking for this one)
- Return a yes/no indicating whether there is a palindrome somewhere contained within the string.
- A minimum-size palindrome is of length 2. For example:
    - whatsup: no
    - radar: yes
    - hello: yes
    - annaareyouok: yes
    - welcometomemphis: yes
    - welcometorhodes: no

You can use any algorithm you like to solve this problem. Following the example in the book, though, your best bet might be to modify the Stack class you wrote for the previous problem (just make it work with chars instead of ints) and also introduce a Queue class.

You don't have to write your own Queue class. Source files are given below, for a Queue of chars. Feel free to use them; they should work well with your Stack class.

- Queue.cpp
- Queue.h

5 points extra credit (do not attempt until you've completed Problem 3):
In addition to determining if a string contains a palindrome, print out what the palindrome is as well as its length:

- whatsup: no
- radar: yes, radar, 5
- hello: yes, ll, 2
- annaareyouok: yes, anna, 4
- welcometomemphis: yes, mem, 3
- welcometorhodes: no