University of Wollongong
School of Computer Science and Software Engineering

CSCI124/MCS9124 Applied Programming Autumn 2015

Lab 10 (1 mark)
Due at 11:59pm on Thurs 21st May.

Aim:
To practice writing a generic class with exceptions.

Task:
In the lecture on collection classes, we looked at writing our own array class and at a deque class.

In this lab, you will write your own stack class. Although you could write a stack using a linked list (push onto the head and pop from the head) we are going to write one based on array.

Here is a definition for the `Stack` class

```
class Stack
{
        public:
                Stack(int size=20);
                Stack(Stack& from);
                ~Stack();

                void push(T val);
                T pop();

                bool isFull();
                bool isEmpty();

        private:
                int n;
                int top;
                T* data;
};
```

It is written in a generic manner, so T can be pretty much any type. You can change the typedef before the definition to set the type of stack, so if you want a stack of characters you can write `typedef char T;`

Your primary task is to write the implementation for the class `Stack`.

First, look at the private data for the class:
The size of the data array is stored in the integer n.
The integer top will hold the index of the topmost element of the stack, so if the stack holds 3 items, the top item will be at index 2. Consider a suitable value for top when the stack is empty.
The pointer data will be a dynamic array, of size n, representing the stack itself.

You will need to implement:

- a constructor, which allocates an array of the given size and sets appropriate values for n and top.
- a copy constructor, which allocates the array and creates an exact copy of an existing stack.
- a destructor, which simply deletes the data array
- a `push` method, which puts the given value on the end of the array and increments top; if the stack is full it should throw an `out_of_range` exception.
- a `pop` method, which returns the top value and decrements top; if the stack is empty it should throw an `out_of_range` exception
- methods `isFull` and `isEmpty` which return true if the stack is full, or is empty respectively

You should also write a simple driver to test the functionality of your stack class. Remember to also test that the correct exceptions are thrown.

Submission:

You are to submit your three files (`stack.h stack.cpp` and `main.cpp`) using the submit program:

```
$ submit –c csci124 –a lab10 –u <username> <files>
```