CPTR-241 HW # 12
Exception handling

The goal of this homework is to explore handling exceptions. Here is a general description of what we wish to do: create a program that implements a stack class and has a main class that calls methods defined in the stack class to add items to the stack or remove items from the stack. Recall that a stack is implemented using an array.

A stack class should have a constructor that creates a stack object of a given size (size in units of what ever data type the stack holds). While the stack class could have overloaded methods to handle more than one data type, begin by writing methods to handle characters. Required methods include push and pop where push places an item on the stack and pop removes an item.

When working with stacks we typically have a variable we call the stack pointer which keeps track of where the last item was placed (in some designs the stack pointer keeps track of the next location to place an item, but we will have it keep track of where the last item was placed). Thus in implementing the push method we will increment the stack pointer and then use the new value as an index into the array.

Errors occur in the push method when the stack pointer becomes greater than the size of the array and an attempt is made to write beyond the end of the array. An error can also occur when the stack pointer becomes less than zero in the pop method.

Create two different stack classes, lets call them stack1 and stack2. The stacks will hold characters.

In the stack1 class, handle errors within the method, i.e. catch it in the method, and pass back to the main a return value that indicates success or failure. The main program will then test the returned value and determine that an error has occurred (and for this homework, print out a message). For the push method, an integer value of 0 should return for success and an integer value of -1 for failure. The pop method is already returning a char, so we can't also return an integer. However, we can return a non-printing character if there is an error. A reasonable character to choose has a numerical value of zero and may be referred to as a null character.

Example of creating a "null" character by type casting an integer zero to type char:

```
char myChar;
myChar = (char) 0;
```

In the stack2 class, handle errors by catching them in the method but re-throwing them and then catching the error in the main.

To get characters to place in the stack, read them from the keyboard as we did in chapter 3 (see pages 64-66 in the text). Characters should be accepted and placed in the stack until a $ character is entered at which point the main program should repeatedly get characters from the stack and print them out until the stack is empty (note: printing out the data is done in the main).