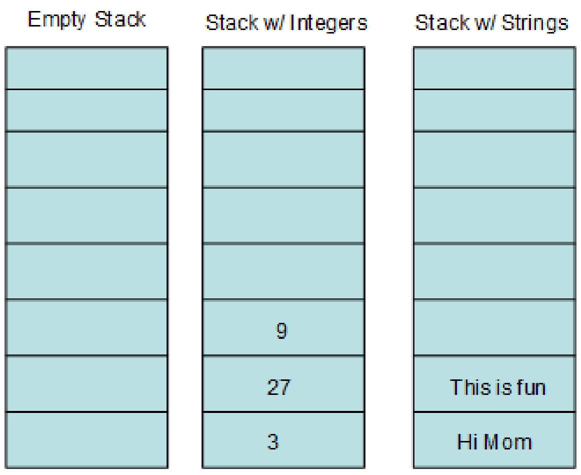
CS 1410 Introduction to Computer Science – CS2
Assignment #8: Exceptions and Template
Total Points: 40 points

Problem Description

Your task is to write **a template class** to implement a unique stack data structure. In addition, the stack operations have to be monitored by using exception handling. That is, you need to protect the stack operations using the appropriate exception handling techniques.

You may read Chapter 18.1 Introduction to the Stack ADT to learn one possible implementation of a stack. The unique stack is a special kind of stack, which has unique items pushed onto it and has items popped off the top. The following diagram illustrates how a unique stack looks and works. There are no duplicated items in this unique stack.



A unique stack may contain any data type. In the example above, the stack contains either integers or strings. A stack could contain floats, doubles, or even other classes (such as the Person class in Assignment 7).

Instead of writing individual classes and functions to handle each possible kind of unique stack, it is useful to **create a generic template unique stack**, define its type in the main code, and allow the compiler to provide the specific implementation at compile time.

Programming Tasks

Create a template unique stack class named TUStack, with at least the following **public member functions**:

TUStack(int nSize): [3 points]

This is the constructor that initializes the maximum size of the stack. In other words, you must not hardcode the size of the stack! Each instance must have **a dynamic stack size** according to the value passed into this constructor.

void Push(T item): [5 points]

This function places a **new, unique** item on the top of the stack. This new item should be different from any of the current items in the stack. For the example shown on the first page, calling the Push function will push the new item right above 9 for the stack with integers and right above "This is fun" for the stack with strings.

T Pop(): [4 points]

This function pops and returns the item on the top of the stack. Before returning, the Pop function should make sure that the second item on the stack is now the first. For the example shown on the first page, calling the Pop function should return 9 for the integer stack and return "This is fun" for the string stack. In the meantime, the top item of the stack would be 27 for the integer stack and "Hi Mom" for the string stack after this operation.

int Size(): [1 point]

This function returns the maximum size of the stack.

int Position(): [1 point]

This function returns the current position of the stack pointer (i.e., the top of the stack).

operator[]: [4 points]

Overload the [] operator so that it returns a copy of the item located at a specified index. This operation is a read-only operation. That is, it does not affect the content in the stack.

[12 points] In addition to the above required public member functions of your class, **you must include exception handling to disallow any error conditions to occur in your code**. Each exception class must allow the calling code (the code that uses the stack class) to obtain **a detailed message about the nature of the error, the current stack size, the current position of the stack pointer, and the new input item if applicable.** Here are a few exceptions you have to consider:

- initialize the maximum size of the stack with a negative value;
- push duplicate items;
- push an item when the stack is full;
- return a copy of the item at an illegal index (e.g., negative index and index above the maximum size of the stack);
- pop value when the stack is empty.

[5 points] The class obviously contains other private data and public functions.

[5 points] Write driver code in your main program that clearly demonstrates all the required elements of this assignment. Specifically, your driver program must test two kinds of stack: a stack with integers and a stack with strings. **Make sure to add appropriate comments to each of your test cases**. It is your responsibility to prove to the grader your code is correct, not for him to guess if it worked correctly.

Good Practice Hint:

1. Design, implement, and test a stack with integers without considering exception handling.
2. Change the stack with integers to a stack with strings without considering exception handling.
3. Add the exception handling.
4. Modify the tested example to be a template.
5. Make sure that there are only two files, TUStack.h and MainProg.cpp. (No TUSTack.cpp!)