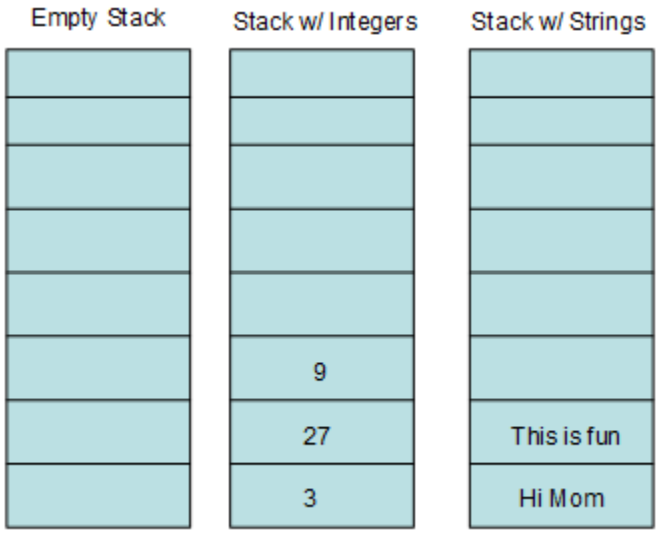
CS 1410 Introduction to Computer Science – CS2
Section 1: MWF 10:30 a.m. – 11:20 a.m.
Section 2: MWF 1:00 p.m. – 1:50 p.m.
Instructor: Xiaojun Qi
Assignment #9: Exceptions and Template

Given: Tuesday, April 1, 2014
Due: 11:59 p.m. Wednesday, April 9, 2014
Total Points: 35 points

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

A unique stack is a simple data structure that 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 Friend class in Assignment 1 and 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.

In this assignment, **you must 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 will initialize 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. (Refer to the posted example Chaper16.SuperArrayException.zip which has a dynamic array size as well.)

void Push(T item): [5 points]

This member function will place 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 Push function will push the new item right above 9 for stack with integers and right above "This is fun" for stack with strings.

T Pop(): [4 points]

This member function will pop and return the item on the top of the stack. For the example shown on the first page, calling Pop function should return 9 for stack with integers and return "This is fun" for stack with strings. In the meantime, Pop function makes sure the item on the top of the stack should be 27 for stack with integers and "Hi Mom" for stack with strings after this operation.

int Size(): [1 point]

This member function will return the maximum size of the stack.

int Position(): [1 point]

This member function will return the current position of the stack pointer (i.e., the top of the stack).

operator[]: [4 points]

Overload the [] operator so that it will return 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.

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 short 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: push duplicate items, push an item when the stack is full, return a copy of the item at an illegal index, and pop value when the stack is empty. **[7 points]**

The class obviously contains other private data and public functions. [8 points]

Write a driver program that clearly demonstrates all the required elements of this assignment. Specifically, your driver program must test two kinds of stack: stack with integers and 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. **[2 points]**

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 Prog9.cpp. (No TUSTack.cpp!)