CS 3358 Assignment 1

Due: 11:59pm, Feb 11, 2025 (Tuesday)

Instructor: Tsz-Chiu Au (chiu.au@txstate.edu)
Doctoral Instructional Assistant: Minhyuk Park (dmz44@txstate.edu)
Grader: Dhruvil Thummar (kwt32@txstate.edu)

This assignment consists of two parts: **stack** and **queue**. Parts of codes are given in the .cpp and .h files in the stack and queue directories. You need to complete the methods or functions in the codes by replacing "`// TODO`" with your own codes. You cannot add new codes or new functions to other parts of the .cpp and .h files.

1. (40%) In `MyStack.cpp`, implement the member functions of the class `MyStack`, which is the class for integer stacks.

2. (20%) In `stackTest.cpp`, complete the implementation of function `postfixTest()`, which uses an integer stack to evaluate post-fix expressions.
   For simplicity, you can assume the post-fix expression is input character by character (i.e., not an entire string), and each input operand is a non-negative, single-digit integer (i.e., 0,1,…,9), while intermediate and final results can be negative and/or multi-digit.
   Your programs must detect invalid/ illegal post-fix expression input, e.g., "4 5 + -".
   Please read the tips in `postfixTest()` for some hints.
   You must put your code inside the while-loop in `postfixTest()` and cannot modify the existing codes or add new functions.

3. (40%) In `MyQueue.h`, implement the queue class **template**, `MyQueue`.
   Note that the `arraySize` needs to be one more than the maximum size of the queue. You cannot add additional fields to the `MyQueue` class.

Both the implementations for the stack and queue must be array-based (i.e., you cannot implement them by linked lists).

Compilation:

You can compile the programs by the following commands on the departmental servers:

```
g++ -o stackTest MyStack.cpp stackTest.cpp
g++ -o queueTest queueTest.cpp
```

Submission:

You should put your implementation of `MyStack.cpp`, `MyStack.h`, `stackTest.cpp` in a folder named `a1_yourNetID`, and then compress the folder into a zip file named `a1_yourNetID.zip`. For example, if your net id is `zz567`, the folder is `a1_zz567`, and the zip file is `a1_zz567.zip`. Finally, please upload the zip file to Canvas. Please do not put other

files in the folder and the zip file.

Sample tests:

The following are a few test cases that you can use to validate your programs. However, passing the following test cases does not guarantee the correctness of your programs. Our grader will evaluate your programs using other test cases, and your program needs to pass them as well.

Sample Output for `stackTest:`

```
Testing the basic functions of the stack...
Please enter the max capacity of the testStack: 3
Testing...
Please enter 'p' for push, 'o' for pop, 'e' for exit:  p
Please enter the integer you would like to push: 5
Please enter 'p' for push, 'o' for pop, 'e' for exit:  p
Please enter the integer you would like to push: 7
Please enter 'p' for push, 'o' for pop, 'e' for exit:  p
Please enter the integer you would like to push: 9
Please enter 'p' for push, 'o' for pop, 'e' for exit:  p
Nothing has been pushed in. The stack is full!
Please enter 'p' for push, 'o' for pop, 'e' for exit:  o
9 has been popped out
Please enter 'p' for push, 'o' for pop, 'e' for exit:  o
7 has been popped out
Please enter 'p' for push, 'o' for pop, 'e' for exit:  o
5 has been popped out
Please enter 'p' for push, 'o' for pop, 'e' for exit:  o
Nothing has been popped out. The stack is empty!
Please enter 'p' for push, 'o' for pop, 'e' for exit:  e
Now, start to use a stack to evaluate postfix expressions...
Please enter the operands (integers 1~9) and operators (+, -, *, /)
one by one...
and enter '=' to indicate the end of the expression and to output
the result.
4
5
7
*
+
2
-
=
The entered post-fix expression results in 37
```

Another run:

```
Testing the basic functions of the stack...
Please enter the max capacity of the testStack: 3
Testing...
Please enter 'p' for push, 'o' for pop, 'e' for exit:  e
Now, start to use a stack to evaluate postfix expressions...
Please enter the operands (integers 1~9) and operators (+, -, *, /)
one by one...
and enter '=' to indicate the end of the expression and to output
the result.
4
5
+
-
Error! No sufficient operands.
```

One more run:

```
Testing the basic functions of the stack...
Please enter the max capacity of the testStack: 3
Testing...
Please enter 'p' for push, 'o' for pop, 'e' for exit:  e
Now, start to use a stack to evaluate postfix expressions...
Please enter the operands (integers 1~9) and operators (+, -, *, /)
one by one...
and enter '=' to indicate the end of the expression and to output
the result.
4
5
+
7
2
-
=
The entered post-fix expression was not a legal one.
```

Sample Output for `queueTest:`

```
Testing the template MyQueue, try an integer queue as an example...
Please enter the max size of the int queue: 2
Please enter 'e' for enqueue, 'd' for dequeue, and 's' for stop.
e
Please enter the integer you want to enqueue: 10
Please enter 'e' for enqueue, 'd' for dequeue, and 's' for stop.
e
Please enter the integer you want to enqueue: 20
Please enter 'e' for enqueue, 'd' for dequeue, and 's' for stop.
e
Cannot enqueue. The queue is full.
Please enter 'e' for enqueue, 'd' for dequeue, and 's' for stop.
d
10 has been popped out.
Please enter 'e' for enqueue, 'd' for dequeue, and 's' for stop.
d
20 has been popped out.
Please enter 'e' for enqueue, 'd' for dequeue, and 's' for stop.
d
Cannot pop. The queue is empty.
Please enter 'e' for enqueue, 'd' for dequeue, and 's' for stop.
s
```