CS 3358 Assignment 1

Due: 11:59pm Thursday, Feb. 20, 2025

This assignment have two parts, under the folder **stack** and **queue**, respectively. Parts of codes are given in the .cpp and .h files. The places you need to fill out in the codes are marked by `//TODO`.

Both the implementations for the stack and queue are supposed to be array-based in this assignment.

1. (20') In `myStack.cpp`, implement the member functions of the class `myStack`, which is the class for integer stacks.

2. (10') In `stackTest.cpp`, complete the implementation of function `postfixTest()`, which uses an integer stack to evaluate postfix 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.
   However, you are supposed to detect invalid/ illegal post-fix expression input, e.g., "4 5 + -".

3. (20') In `myQueue.h`, implement the queue class **template**, `myQueue`.
   Keep in mind, the `arrayLength` needs to be one more than the capacity of the queue.
   Also, under this implementation, make sure your calculation of `currentSize` is correct, and the conditions for "Full" and "Empty" are correct. One shortcut could be: once you make sure `currentSize()` is implemented correctly, you might use it in `isFull()` and `isEmpty()`, and the number of elements in the queue must range from `0` to `arrayLength-1`.

Compiling:

For the **stack** part, you will need to compile `myStack.cpp` and `stackTest.cpp` `separately` to two .o files and then link them together to the executable.

For the **queue** part, you can directly compile `queueTest.cpp` to the executable. Two PDF guides in TRACS might be helpful, if you are not familiar with these in the Linux environment using the g++ compiler.

Submission:

You should submit your work via canvas.

You should put `myStack.cpp, myStack.h, stackTest.cpp` into the folder `/stack`, put `myQueue.h, queueTest.cpp` into the folder `/queue`, and pack the folders `/stack` and `/queue` into a single .zip file to upload to TRACS. The .zip file should be named as a1_yourNetID.zip, such as a1_zz567.zip

Sample tests:

Note that successes in getting the following test results do not guarantee the correctness of your work and therefore do not guarantee you a satisfactory grade, whereas failures in getting the following test results probably do indicate flaws in your work and you may lose points.

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
```