EECS 560 Lab 3: Stacks and Queues

Due date:
09/18/2020, 11:59 pm -Friday

Objective
Get familiar with Stack and queue ADT implementation using array with C++.Work on basic operations of Stack and Queue.

Specification of the Stack ADT Requirements:

1. Create a Stack class with name **"myStack".** Which is templated and Array based implementation.

2. Implement **constructor** with default behavior as **myStack(void):** initialize an empty stack with capacity of 10

3. **destructor** with default behavior .

4. Implement a **deep copy constructor** with a move constructor and move assignment operator. The constructor should have an interface of **myStack(myStack && rhs)** and move assignment operator **myStack & operator= (myStack && rhs)**

5. Implement a **copy constructor** with copy constructor and copy assignment operator. The constructor should have an interface of **myStack(const myStack<Object> & rhs)** and the copy assignment operator **myStack & operator= (const myStack & rhs)**

6. **push():** To add an element to stack. Stack capacity should be monitored and should be handled internally if full. capacity should be doubled if stack is full.

7. **pop():** Remove an element from stack. This method should return the removed element

8. **top():** This should return the top element of the stack

9. **empty() :** This method to verify if stack is empty. Returns 0 if empty 1 if not empty.

10. **clear()** : This method to clear elements of stack.

Specification of the Queue ADT Requirements:

1. Create a Queue class with name **"myQueue".** Which is templated and Array based implementation.

2. Implement **constructor** with default behavior as **myQueue(void):** initialize an empty queue with capacity of 10

3. **destructor** with default behavior .

4. Implement a **deep copy constructor** with a move constructor and move assignment operator. The constructor should have an interface of **myQueue (myQueue && rhs)** and move assignment operator **myQueue & operator= (myQueue && rhs)**

5. Implement a **copy constructor** with copy constructor and copy assignment operator. The constructor should have an interface of **myQueue(const myQueue<Object> & rhs)** and the copy assignment operator **myQueue & operator= (const myQueue & rhs)**

6. **enqueue():** To add an element to queue. Queue capacity should be monitored and should be handled internally if full. capacity should be doubled if queue is full.

7. **dequeue():** Remove an element from queue. This method should return the removed element

8. **front():** This should return the front element of the queue

9. **empty() :** This method to verify if queue is empty. Returns 0 if empty 1 if not empty.

10. **clear()** : This method to clear elements of queue.

Testing and Grading

We will test your implementation using a tester main function, on a number of instances that are randomly generated. We will release the tester main function, several instances (will be different from the grading instances but will have the same format), and expected output together with the lab instruction via Blackboard. You code will be compiled under Ubuntu 20.04 LTS using g++ version 9.3.0 (default) with C++11 standard.

The command line we are going to use for compiling your code is:

"g++ -std=c++11 main.cpp" (note that main.cpp will try to include the .hpp files you submit, and your .hpp files needs to be property implemented to compile successfully).

Your final score will be the percentage your program passes the grading instances. Note that if your code does not compile (together with our tester main function), you will receive 0. Therefore, it is very important that you ensure your implementation can be successfully compiled before submission.

Submission and Deadline

Please submit your implementation as two different .hpp files, with file names "myStack_[YourKUID].hpp" and "myQueue_[YourKUID].hpp". For example, if my KU ID is c123z456, my submission will be a single file named "myStack_c124z456.hpp" and "myQueue_c124z456.hpp". Submissions that do not comply with the naming specification will not be graded. Please submit through Blackboard.