Mariya Eggensperger
CST 370, Spring 2017
Dr. Feiling Jia
Design/Analysis
of Algorithms

CST 370 Homework (Stacks and Queues)

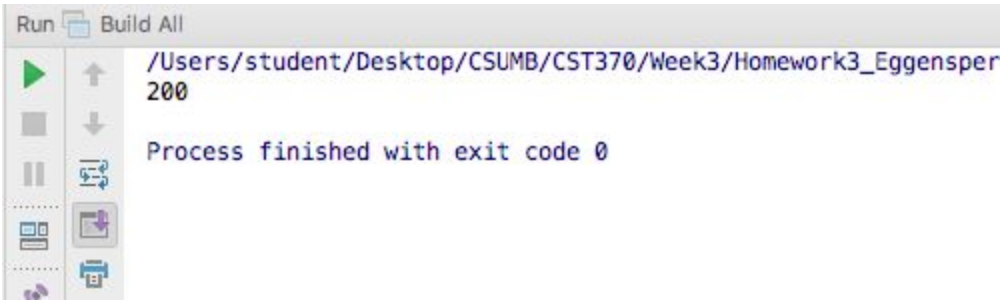
1. (20 points) Convert the following infix expressions to postfix expressions
a) (2+3)*6+5*6-7 infix to **postfix 2 3 + 6 * 5 6 * + 7 -**
b) 2+3*7+(4-6*7) infix to **postfix 2 3 7 * + 4 6 7 * - +**

2. (20 points) In the following code, assume the myQueue object is a queue that can hold integers. (The lines are numbered for reference purposes.)

```
1. myQueue.enqueue(200);
2. myQueue.enqueue(100);
3. myQueue.enqueue(300);
4. cout << myQueue.front() << endl;
5. myQueue.dequeue();
6. myQueue.dequeue();
7. cout << myQueue.front() << endl;
```

What will the statement in line 4 display? _____

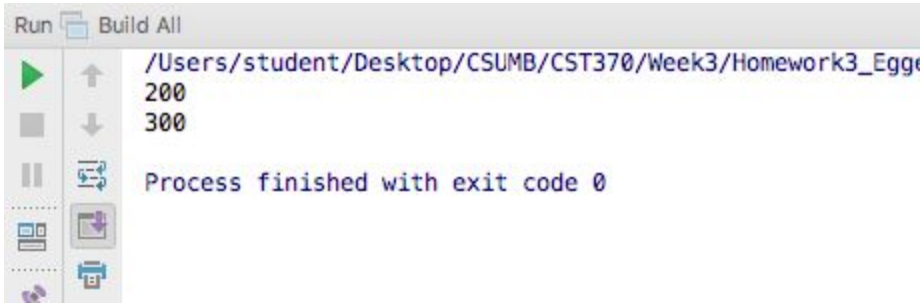
On line 4, the program will display the ‘front’ element in the queue, which when enqueued, is the element 200. See image.



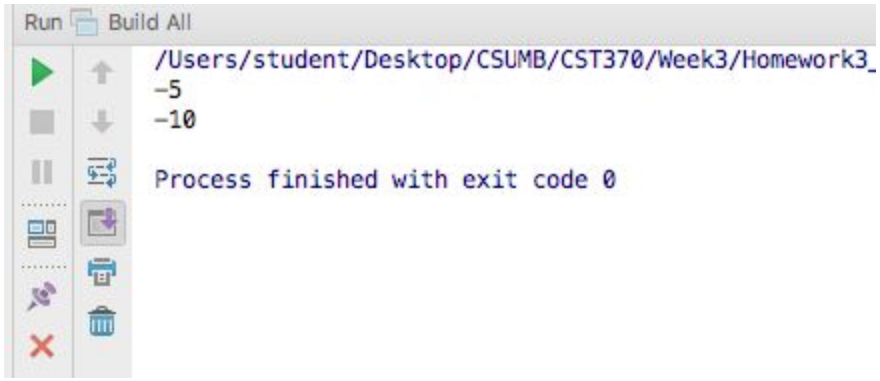
What will the statement in line 7 display? _____

On line 7, since the queue is nonempty, the program will display the retrieved value(s) from the front(s) (elements: 200, 300) of the queue. See image.

It is noteworthy that, the value 200 comes from line 4. cout << myQueue.front() << endl and the value 300 comes from line 7. cout << myQueue.front() << endl;



3. (20 points) Enqueue 5 numbers [6, 3, 1, -5, -10] in order. Then dequeue 3 elements from the queue. Print out contents of the current queue.



4. (40 points) Write an algorithm to implement a stack using two queues (say q1 and q2). Specifically, you need to implement the pop() and push() functions of a stack. You can assume that you have the implementation of the queue available and you can use the enqueue() and dequeue() functions of the queue. Note stack is a LIFO data structure while queue is a FIFO data structure.

Though a pseudocode or code will be preferred you will still be given points if you describe the algorithm in plain English as a sequences of steps. For example, while writing the pseudocode if you want to call the enqueue() function for queue q1, you can call it as q1.enqueue().

PSEUDOCODE answer:
***Please read comments for pseudocode narration.**

```
void push(int)
{
  q1.enqueue(x); // Push a value into q1
  q2.enqueue(y);  // Push a value into q2

}

int pop()
{
  if(q1.empty() && q2.empty) // If q1 and q2 stacks are empty
  {
    throw StackException("StackException: stack empty on pop"); // Throw an empty stack error exception
  }
  if(!q1.empty()) // If not q1 is empty
  {
    // Then, while the size of q1 is greater than 1
    while(q1.size()>1)
    {
      // Do the following to swap
      (q2.enqueue(q1.dequeue)); // Add the dequeued value from q1 to q2
      return q1.dequeue; // Return the dequeued value of q1
    }
    while (q2.size()>1) // While the size of q2 is greater than 1
    {
      // Do the following to swap
      q1.enqueue(q2.dequeue()); // Add the dequeued value from q2 to q1
      return q2.dequeue(); // Returned the dequeued value of q2
    }
  }
}
```