CST 370
Homework (Stacks and Queues)

1. Convert the following infix expressions to postfix expressions

a) (2+3)*6+5*6-7

23+6*56*+7-

b) 2+3*7+(4-6*7)

237*+467*-+

2. 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? ______200_______________

What will the statement in line 7 display? ______300______________

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

-5, -10

4. 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().

First define a stack as having a member called q1, a queue.

Note: dequeue() function returns the value of the removed item.

```
push(val)
    {
      new queue = q2 //create local empty queue for utility

      //if stack is empty, simply enqueue() val and return
      if (q1.front() == NULL)
      {
         q1.enqueue(val) //add new value and return
         return
      }

      //else q1 already contains value(s). move values to q2,
      //enqueue() the new value to q1, and then move values from q2
      //back to q1
      else
      {
         //loop until q1 is empty. Transfer q1 dequeue'd value to q2
         while (q1.front() != NULL)
         {
            temp = q1.dequeue()
            q2.enqueue(temp)
         }

         //add the new value to q1
         q1.enqueue(val)

         //loop until q2 is empty. Transfer q2 dequeue'd value to q1
         while (q2.front() != NULL)
         {
            temp = q2.dequeue()
            q1.enqueue(temp)
         }
      }
    return //q1 now updated
    }

pop()
{
   //dequeue() will remove the most recent value pushed
   q1.dequeue()
   return;
}
```