QUEENS COLLEGE Department of Computer Science
CSCI 313 Practice problems on Stacks and Queues

Instructor: Krishna Mahavadi

These are some of the solutions to practice problems. Not all problems have solutions here.

Solutions to some older problems might not make use of generics. Generics are now required in this course.

Problem 1 The generic class Queue is to be programmed with an array based implementation. A partial version of the implementation follows. The two most important methods have been omitted. The treatment of an empty Queue (as given by the constructor) is different from the one we used in class. Make sure that your methods work correctly with the choices made by the constructor.

```
public class P2<T> implements Queue<T> {
   private T data[];
   private int front, rear, size;
   public P2() { data = (T[]) new Object[100]; front = size = 0; rear = -1; }
   public int size() { return size; }
   public boolean empty() { return size == 0; }
   // methods omitted here
}
```

(a) **Identify the two missing methods**. For each give the name, parameters and return type.

Answer:

```
   public T dequeue() throws Exception
   public void enqueue(T x) throws Exception
```

(b) **Give a complete implementation** of **ONE** of the two missing Queue methods. (You can choose either of them.)

Answer:

```
   public T dequeue() throws Exception {
      if (size() <= 0) throw new Exception("Queue Empty");
      size--;
      T ans = data[front++];
      if (front == 100) front = 0;
      return ans;
   }

   public void enqueue(T x) throws Exception {
      if (size() >= 100) throw new Exception("Queue Full");
      rear++;
      if (rear == 100) rear = 0;
      data[rear] = x;
      size++;
   }
```

Problem 2 (i) Write the Java interface for the ADT Stack. (Just write an interface that specifies methods — **do not** implement the methods.)

Answer:

```
interface Stack {
   public Object pop();
   public void push(Object x);
   public int size();
   public boolean empty();
}
```

(ii) Write the Java interface for the ADT Iterator. (Just write an interface that specifies methods — **do not** implement the methods.)

Answer:

```
interface Iterator {
   public Object next();
   public boolean hasNext();
   public void remove();
}
```

(iii) Write a client function with title line:

```
public static int count(Stack s, Object x)
```

The function returns a count of the number of times the object x is found on the Stack s. When the method ends the Stack should store exactly the same data as at the beginning of the method. However your function will need to alter the Stack data as it runs.

Answer:

```
   public static int count(Stack s, Object x) {
     Stack temp = new Stack();
     int ans = 0;

     while (!s.empty()) {
        Object y = s.pop();
        if (y.equals(x)) ans++;
        temp.push(y);
     }
     while (!temp.empty()) s.push(temp.pop());
     return ans;
   }
```

(iv) Write a client function with title line:

```
public static int count(Iterator i, Object x)
```

The function returns a count of the number of times the object x is found on the Iterator i. This function can and should empty out the Iterator and should not restore it to its original state.

Answer:

```
   public static int count(Iterator i, Object x) {
     int ans = 0;
     while (i.hasNext()) {
        Object y = i.next();
        if (y.equals(x)) ans++;
     }
     return ans;
   }
```