CS 2510 Exam 3 – Spring 2012

Name: _____

Student Id (last 4 digits): _____

- Write down answers in the space provided.
- You may use all syntax you know from class (that is, the parts of Java we have studied in class), and you may use any Java Collection Framework classes you know how to use.
- When defining methods, you do not need to give a complete class definition—just indicate in which class your method definition should be placed.
- For tests you only need to provide the expression that computes the actual value, connecting it with an arrow to the expected value. For example `s.method() -> true` is sufficient.
- Remember that the phrase "design a class" or "design a method" means more than just providing a definition. It means to design them according to the **design recipe**. You are *not* required to provide a method template unless the problem specifically asks for one. However, be prepared to struggle if you choose to skip the template step.
- A word of advice: Decompose complex problems into small steps.
- We will not answer *any* questions during the exam.

Good luck!

Problem	Points	/
A		/10
B		/15
C		/10
Total		/30

10 Points

Problem 1

Design the `merge` method of the the `MergeAlgorithm` class, whose purpose is to take two sorted lists (both are assumed to be ordered by a given comparator) and to produce a single merged list that contains all the elements of the two lists in sorted order.

```
import java.util.*;
import tester.*;

class MergeAlgorithm {

  <T> ArrayList<T> merge(ArrayList<T> list1,
                         ArrayList<T> list2,
                         Comparator<T> comp) {
        ...
  }
}
```

The `merge` method should not mutate any of its arguments.

Solution

```
class Merge {

    // Merge two sorted lists into a single sorted list.
    // ASSUME: list1, list2 are sorted according to comp.
    <T> ArrayList<T> merge(ArrayList<T> list1,
   ArrayList<T> list2,
   Comparator<T> comp) {

ArrayList<T> result = new ArrayList<T>();
Integer i1 = 0;
Integer i2 = 0;

while (!i1.equals(list1.size()) &&
       !i2.equals(list2.size())) {

    if (comp.compare(list1.get(i1), list2.get(i2)) < 0) {
result.add(list1.get(i1));
```

```
i1 = i1 + 1;
    } else {
result.add(list2.get(i2));
i2 = i2 + 1;
    }
}
while (!i1.equals(list1.size())) {
    result.add(list1.get(i1));
    i1 = i1 + 1;
}
while (!i2.equals(list2.size())) {
    result.add(list2.get(i2));
    i2 = i2 + 1;
}

return result;
    }
}


class CharComp implements Comparator<Character>{
  public int compare(Character c1, Character c2){
    return c1.compareTo(c2);
  }
}

class ExamplesMerge{
  ExamplesMerge(){}

  ArrayList<Character> alphabet1 =
      new ArrayList<Character>(Arrays.asList(
          'a', 'c', 'e', 'g', 'i',
          'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's'));

  ArrayList<Character> alphabet2 =
      new ArrayList<Character>(Arrays.asList(
          'b', 'd', 'f', 'h', 'j',
          't', 'u', 'v', 'w', 'x', 'y', 'z'));

  ArrayList<Character> alphabet =
```

```
      new ArrayList<Character>(Arrays.asList(
          'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
          'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's',
          't', 'u', 'v', 'w', 'x', 'y', 'z'));

  ArrayList<Character> result =
      new ArrayList<Character>();

  CharComp comp = new CharComp();
  Merge ma = new Merge();

  void testMerge(Tester t){
    t.checkExpect(ma.merge(alphabet1, alphabet2, comp, result), alphabet);
  }
}
```

[Here is some more space for the previous problem.]

15 Points

Problem 2

Class `CircularList` is similar to the `Dequeue` you have designed in your homeworks, but has only one head node and the last node in the list is connected back to the head - making the list circular.

```
// to represent one node in a circular list
class Node<T>{
  T data;
  Node<T> prev;
  Node<T> next;

  Node(T data, Node<T> prev, Node<T> next){
    this.data = data;
    this.prev = prev;
    this.next = next;
  }

}

// to represent a circular list
class Circular<T>{
  Node<T> ht;

  Circular(){
    this.ht = new Node(null, null, null);
    this.ht.prev = this.ht;
    this.ht.next = this.ht;
  }
...
}
```

Design the following methods for this circular list:

A. Design the method `isEmpty()` for the `Circularlist`.

Solution

```
// is this list empty?
public boolean isEmpty(){
  return(this.ht.next == this.ht);
}
```

B. Design the method `void addAtFront(T t)` for the `Circularlist` that adds the node with given item as its value at the front of this list.

Solution

```
// add the given item to the front of this list
public void addAtFront(T t){
  Node<T> oldLast = this.ht.next;
  Node<T> n = new Node<T>(t, this.ht, oldLast);
  this.ht.next = n;
  oldLast.prev = n;
}
```

C. Design the method `void addAtTail(T t)` for the `Circularlist` that adds the node with given item as its value at the tail of this list.

Solution

```
//add the given item to the tail of this list
public void addAtTail(T t){
  Node<T> oldLast = this.ht.prev;
  Node<T> n = new Node<T>(t, oldLast, this.ht);
  this.ht.prev = n;
  oldLast.next = n;
}
```

D. Design the method `T removeFromFront()` for the `Circularlist` that removes the node at the front and returns its value.

Solution

```
//remove the given item from the front of this list
public T removeFromFront(){
  if (this.isEmpty())
    throw new RuntimeException("Cannor remove from an empty list");
  else{
    Node<T> removed = ht.next;
    ht.next = removed.next;
    removed.next.prev = ht;
    return removed.data;
  }
}
```

E. Design the method `T removeFromTail()` for the `Circularlist` that removes the node at the tail and returns its value.

Solution

```
//remove the given item from the tail of this list
public T removeFromTail(){
  if (this.isEmpty())
    throw new RuntimeException("Cannot remove from an empty list");
  else{
    Node<T> removed = ht.prev;
    ht.prev = removed.prev;
    removed.prev.next = ht;
    return removed.data;
  }
}
```

[Here is some more space for the previous problem.]

Solution

```
class ExamplesCircular{
  ExamplesCircular(){}

  Circular<String> cir = new Circular<String>();

  void testCircular(Tester t){
    t.checkExpect(this.cir.isEmpty(), true);
    //cir.addAtFront("hello");
    //cir.addAtFront("ciao");
    //cir.addAtFront("bye");

    cir.addAtTail("hello");
    cir.addAtTail("ciao");
    cir.addAtTail("bye");

    System.out.println("Adding at the tail:");

    System.out.println(this.cir.ht.next.data);
    System.out.println(this.cir.ht.next.next.data);
    System.out.println(this.cir.ht.next.next.next.data);

    System.out.println(this.cir.ht.prev.data);
    System.out.println(this.cir.ht.prev.prev.data);
    System.out.println(this.cir.ht.prev.prev.prev.data);

    t.checkExpect(this.cir.isEmpty(), false);
    t.checkExpect(this.cir.removeFromTail(), "bye");
    t.checkExpect(this.cir.removeFromTail(), "ciao");
    t.checkExpect(this.cir.removeFromFront(), "hello");

    t.checkExpect(this.cir.isEmpty(), true);

    cir.addAtFront("hello");
    cir.addAtFront("ciao");
    cir.addAtFront("bye");

    System.out.println("Adding at the front:");
```

```
    System.out.println(this.cir.ht.next.data);
    System.out.println(this.cir.ht.next.next.data);
    System.out.println(this.cir.ht.next.next.next.data);

    System.out.println(this.cir.ht.prev.data);
    System.out.println(this.cir.ht.prev.prev.data);
    System.out.println(this.cir.ht.prev.prev.prev.data);

    t.checkExpect(this.cir.isEmpty(), false);
    t.checkExpect(this.cir.removeFromFront(), "bye");
    t.checkExpect(this.cir.removeFromFront(), "ciao");
    t.checkExpect(this.cir.removeFromTail(), "hello");

    t.checkExpect(this.cir.isEmpty(), true);
  }
}
```

10 Points

Problem 3

You are now masters of encoding and decoding algorithms. Here is a new one for you to implement: When encoding the letters, leave the first letter the same as the original. Replace the second letter of the message with a letter that is one space later in the alphabet. (So, if the second letter was 'e', you would replace it with 'f'.) Replace the third letter with one that is two letters later in the alphabet. So, if the third letter was 'l', you would replace it with 'n'.) Proceed like this till the end. Of course, circle around to the beginning of the alphabet, if you get to the end. (So, if 'w' appears as the 7th letter, it will be replaced by 'b' — we included a 27th character, the space.)

For example, "hello world" would be encoded as "hfnosebvzun".

We are giving you the `ArrayList<Character>` that contains all letters of the alphabet and the space character at the end:

```
interface DTree {
  /** The original list of characters used in encoding and decoding */
  ArrayList<Character> alphabet =
        new ArrayList<Character>(Arrays.asList(
            'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
            'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's',
            't', 'u', 'v', 'w', 'x', 'y', 'z', ' '));
}
```

A. Design the method `encode` that consumes a message `String` and produces the encoded `String`

Solution

```
class Encoder implements DTree{
  Encoder(){}

  String encode(String source, String acc){
    if (source.length() > 0)
    return encode(source.substring(1),
        acc + alphabet.get((alphabet.indexOf(source.charAt(0)) + acc.length()) %
    else
      return acc;
  }


  String decode(String source, String acc){
    if (source.length() > 0)
    return decode(source.substring(1),
        acc + alphabet.get((alphabet.indexOf(source.charAt(0)) - acc.length()+ 2
    else
      return acc;
  }

}
```

B. Design the method `decode` that consumes an encoded message `String` and produces the original message `String`

Solution

```
class ExamplesExam3{
  ExamplesExam3(){}

  Encoder enc = new Encoder();

  void testEncoder(Tester t){
    t.checkExpect(enc.encode("hello world", ""), "hfnosebvzun");
    t.checkExpect(enc.encode("how are you", ""), "hpycewkgfxd");

    t.checkExpect(enc.decode("hfnosebvzun", ""), "hello world");
    t.checkExpect(enc.decode("hpycewkgfxd", ""), "how are you");
  }

  public static void main(String[] argv){
    ExamplesExam3 e = new ExamplesExam3();
    Encoder enc = new Encoder();

    System.out.println("hello world" + " encoded as " +
        enc.encode("hello world", ""));
    System.out.println("how are you" + " encoded as " +
        enc.encode("how are you", ""));

    System.out.println("hello world" + " decoded as " +
        enc.decode("hfnosebvzun", ""));
    System.out.println("how are you" + " decoded as " +
        enc.decode("hpycewkgfxd", ""));

    Tester.runFullReport(e);
  }
}
```