UC Berkeley – Computer Science
CS61B: Data Structures

Midterm #1-Redemption, Spring 2016

This test has 3 questions worth a total of 14 points, though your overall score is capped at 10 points. The exam is closed book, except that you are allowed to use a one page written cheat sheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. Write the statement out below, and sign once you're done with the exam. **Write the statement out below in the blank provided and sign. You may do this before the exam begins.**

"I have neither given nor received any assistance in the taking of this exam."

____

____

Signature: ______________________

	Points		Points
1	3		
2	5		
3	6		

Name:
SID:
Three-letter Login ID:
Login of Person to Left:
Login of Person to Right:

Exam Room: ________________

Total	10

Tips:

- There may be partial credit for incomplete answers. Write as much of the solution as you can, but bear in mind that we may deduct points if your answers are much more complicated than necessary.

Optional. Mark along the line to show your feelings on the spectrum between ☹ and ☺.

Before exam: [☹________________☺].
After exam: [☹________________☺].

1. **References (3 Points).**

a) Suppose we are given the following `StringList` class:

```
public class StringList {
   public String val;
   public StringList tail;
   public StringList(String val, StringList tail) {
       this.val = val;
       this.tail = tail;
   }
   public String toString[1]() {
       if (tail == null) { return val; }
       return val + “ “ + tail.toString();
   }
}
```

What will be printed by the code below? Write your answer in the three blanks provided.

```
public static void nullify(StringList list) {
    list.tail.tail = null;
}

public static void nullify2(StringList list) {
    StringList temp = list;
    temp.tail = null;
}

public static StringList nullify3(StringList list) {
    list = null;
    return list;
}

public static void main(String[] args) {
    StringList L = new StringList(“we”, null);
    L.tail = new StringList(“love”, null);
    L.tail.tail = new StringList(“pie”, null);
    System.out.println(L);         _______we love pie________
    nullify(L);
    System.out.println(L);         __________________________
    nullify2(L);
    System.out.println(L);         __________________________
    nullify3(L);
    System.out.println(L);         __________________________
}
```

[1] As discussed in lecture on 2/24/2016, this method is automatically called if you print an object.

2. **Array Processing (5 Points).** Write a method `rotateLeft(char[] input, int K)` that **non-destructively** returns a version of `input` that is rotated to the left by `K` spaces. You may not need all lines. You may assume `K > 0.` Partial credit will be given for code that cannot handle `K > input.length`.

For example, if `input` is `['a', 'b', 'c', 'd']`,

- and `K` is 0, this method will return `['a', 'b', 'c', 'd']`.
- and `K` is 1, this method will return `['b', 'c', 'd', 'a']`.
- and `K` is 3, this method will return `['d', 'a', 'b', 'c']`.
- and `K` is 5, this method will return `['b', 'c', 'd', 'a']`.

```
public static char[] rotateLeft(char[] input, int K) {
    ________________________________________________
    ________________________________________________
    ________________________________________________
    ________________________________________________
    ________________________________________________
    ________________________________________________
    ________________________________________________
    ________________________________________________
    ________________________________________________
    ________________________________________________
    ________________________________________________
}
```

3. Interfaces (6 points). You may assume that you have a correct implementation of `ArrayIntDeque` and `LinkedListIntDeque` throughout this problem, and that each implements the `IntegerDeque` interface. These classes are just like the proj1a classes but only support integers (i.e. are not generic).

a) Consider the `IntegerDeque` interface below. It is exactly like the `Deque` interface from proj1c, but is for ints only. Add a default method `xify` that replaces each item with itself x times in the `IntegerDeque`. You may assume all numbers are ≥ 0. For example:

- If the `IntegerDeque` contains `[3, 1, 0, 2, 2]`, then the `IntegerDeque` should contain `[3, 3, 3, 1, 2, 2, 2, 2]` after `xify()` is called.
- Order matters. Your answer for our example **<u>should not be</u>** `[3, 1, 2, 3, 3, 2, 2, 2]`.

```
public interface IntegerDeque {
        void addFirst(int x);                  void addLast(Item x);
        boolean isEmpty();                     int size();
        void printDeque();                     int get(int index);
        int removeFirst();                     int removeLast();
    default void xify() {
         __________________________________________
         __________________________________________
         __________________________________________
         __________________________________________
         __________________________________________
         __________________________________________
         __________________________________________
         __________________________________________
         __________________________________________
    }
}
```

For example the test below should pass.

```
IntegerDeque ad = new ArrayIntDeque();
ad.addLast(3);
ad.addLast(1);
ad.addLast(0);
ad.addLast(2);  /* After this line, ad is: [3, 1, 0, 2] */
ad.addLast(2);  /* After this line, ad is: [3, 1, 0, 2, 2] */
ad.xify();      /* After this line, ad is: [3, 3, 3, 1, 2, 2, 2, 2] */
assertEquals(3, ad.get(1));
```