CS II: Data Structures
Lab 4: Exam 1 Review

Here are some more practice questions to get you prepared for the exam!

Question 1

Consider the following code:

```
public class ABC {

    private static int count = 0;
    private int x;

    public ABC(int i) {
        x = i;
    }

    public void incrementCount() {
        count++;
    }

    public void printX() {
        System.out.println("Value of x : " + x);
    }

    public static void printCount() {
        System.out.println("Value of count : " + count);
    }

    public static void main(String[] args) {
        ABC demo = new ABC(5);

        ABC Obj1 = new ABC(2);
        ABC Obj2 = new ABC(5);

        Obj1.printX();
        Obj1.incrementCount();
        demo.incrementCount();
        Obj1.printCount();
        Obj2.printCount();
        Obj2.printX();
        Obj1.incrementCount();
        Obj1.printX();
        Obj1.printCount();
        Obj2.printCount();
    }
}
```

List what will be printed on screen:

Question 2

The `StringLinkList` class describes a string of characters represented as a singly linked list. Consider the following method added to the `StringLinkList` class:

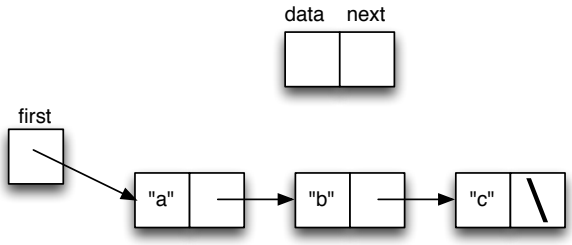

Figure 1

```
void mystery() {
        StringLinkList newFirst = first;
        while (first.next != null) {
                StringLinkList temp = first.next;
                first.next = temp.next;
                temp.next = newFirst;
                newFirst = temp;
        }
    }
```

Suppose that the above `mystery` method is applied to a `StringLinkList` object containing three strings, "a", "b", and "c" in that order. (See the figure above.)

Show how this `StringLinkList` object changes during each step of the method's execution. Specifically, show the structure of the `StringLinkList` object immediately after each execution of the statement `newFirst = temp`. Make sure you show exactly where `temp`, `first`, and `newFirst` are pointing.

Answer.

Question 3

Assume that we have a linked list class without a sentinel node.

```
class LinkedList {
    private ListNode head;

    public LinkedList() { head = null; }

   class ListNode{
       int data;
       ListNode next:
   }

   private void mystery2( ListNode t ){

     if(t == null)
          return;

     mystery2( t.next )
     System.out.println(t.data);
   }

   public void foo() {
     mystery(head);
   }
...
}
```

***What does the foo()* method do? Be specific.**

***What would the foo()* method do if we changed the following line of code?**

~~`if(t == null)`~~ `if(t.next == null)`

Question 4

Write a ***recursive*** function that takes a key from the user, and return the position of the element in the linked list if it exists; and it returns -1 if element is not in the list.
(assume the same `LinkedList` class listed in Q3). Note that we need a helper method with different parameters. You must fill in `searchIndex` and `searchIndexHelper`.

Answer:

```
class LinkedList {
...
      public int searchIndex( int key ) {




      }
      private int searchIndexHelper(ListNode t, int key) {









      }
}
```

Question 5

Consider the following code:

```
int[] A={  ,  ,  ,  ,  ,  ,  };
int i=0;
int j=6;

boolean result = true;
while( i<=j ){
     if(A[i] != A[j])
          result false;

     i++;
     j--;
}

System.out.println(result);
```

Array A is not initialized. Fill A with numbers such that:
the result will be “true”:

A ={ , , , , , , }

the result will be “false”:

A ={ , , , , , , }