CS272 Final
Time: 10:30am-12:30pm, Tue. Dec. 9, 2014; SH113
Close book; Points: 100pts

Banner ID: ________________________ Name: ___________________________ Score: _____

Q1. (10 pts) Given the code for the class *MyBag*, please write down the content of *bag1* (*num*, the capacity, and the real content in *data*) after running the main function.

```
public class MyBag<E> {
    public E[] data=null;
    public int num=0;
    public MyBag(){data = (E[])new Object[2];  num = 0; }

    private void ensureCapacity(int minimumCapacity){
        if(minimumCapacity <= data.length) return;
        else{
                E[] newData =  (E[])new Object[minimumCapacity];
                System.arraycopy(data, 0, newData, 0, num);
                data = newData;
        }
    }
    public void add(E a){
        if(num==data.length) ensureCapacity((data.length+1)*2);
        data[num++] = a;
    }

    public boolean f(E a) {
        if(a==null) return false;
        int i=0, oldnum = num;
        while(i<num){
                if(data[i].equals(a)) {data[i]=data[num-1]; num--;}
                else i++;
        }
        if(oldnum==num) return false;
        else return true;
    }

    public static void main(String[] args) {
        MyBag<Integer> bag1 = new MyBag<Integer>();
        bag1.add(2); bag1.add(3); bag1.add(2); bag1.add(4); bag1.add(5);
        bag1.f(2);
    }
    Result: ___________________________________________________________
}
```

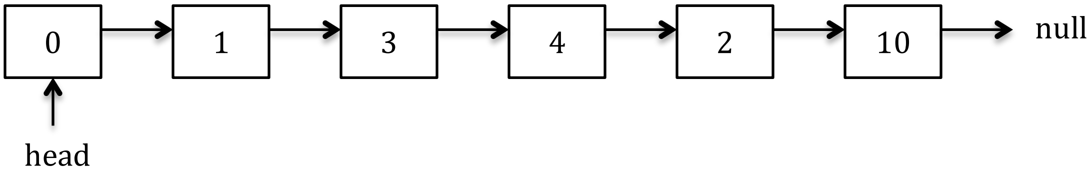
Q2. (15 pts) **(Linked list)** Given the *SNode* class as follows.
A. (5 pts) Finish the ***size*** method for the class *SNode*.

```
public class SNode <E>{
    public E data;
    public SNode<E> next = null;
    public SNode(){; }

    //A method to get the number of nodes in the list starting from a given node head.
    public int size(SNode<E> head) {






    }

    public static SNode f (SNode head){
        SNode cursor = head;
        SNode prev = null;
        SNode next = null;

        while(cursor!=null){
                next = cursor.next;
                cursor.next = prev;
                prev = cursor;
                cursor = next;
        }

        head = prev;
        return head;
    }
}
```

B. (10 pts) What is the worst-case complexity of the **above f method** in Big-O ____________

define n= ______________

Given the above function *f()*, show the result of running *f(head)* on a given list shown as follows.



Q3. (10 pts) **[Stack and Queue]** Answer the following questions by utilizing the *SNode* class given above.

A. (5 pts) Implement an O(1) ***push*** method for the class *LinkStack*, this method needs to match the pop method.

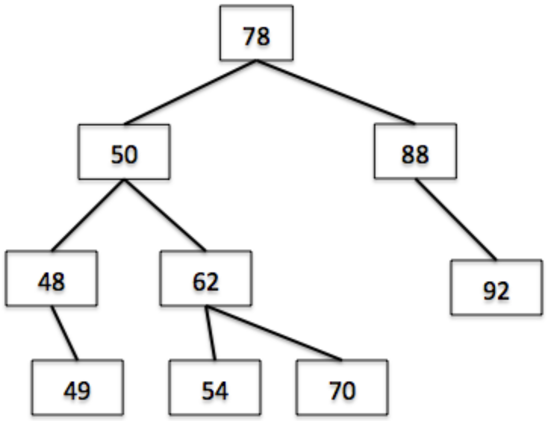
B. (10 pts) Implement an O(1) ***enqueue*** method for the class *LinkedQueue*.

```
public class LinkStack<E> {
    public SNode<E> top;
    public LinkStack()  {top = null;}

    public void push(E e) {//Insert data to the stack









    }

    public E pop() {
        if(top==null) throw new EmptyStackException();
        E answer = top.data;
        top = top.next;
        return answer;
    }
}

public class LinkedQueue<E> {

    public SNode<E> rear = null;                    //the rear of a queue
    public SNode<E> front = null;                   //the front of a queue
    public LinkedQueue(){; }

    public void enqueue(E e) {//insert data to the queue rear









    }
}
```

Q4. (30 pts) [**Binary search tree and AVL tree**] Given the classes *AVLNode* and *AVL* as follows. Assume duplication values are not allowed in the tree, please
A. (10 pts) Finish the **searchRecursion** method in the *AVL* class.
B. (10 pts) Design **avg()** function to calculate the average value of all the values in an AVL tree. You can design and use facilitating functions when necessary.

```
class AVLNode{
        public int data;          //the element value for this node
        public AVL left;          //the left child of this node
        public AVL right;         //the right child of this node
        public int height=1;      //height of the tree rooted at this node

        public AVLNode()          {data = 0; left = new AVL(); right = new AVL(); }
        public AVLNode(int initData){data = initData; left = new AVL();right = new AVL();}

}

public class AVL {
        public AVLNode            root; //instance variable to denote the root of the AVL tree
        public AVL()              {root = null;}

        public AVLNode searchRecursion(int e){//search e recursively



        }
        public int avg(){



        }
}
```

Given the following AVL tree:



C. (5pts) Assume that you are required to remove the value "50" from the above AVL tree using the *remove* algorithm that we discussed in class. Draw the AVL tree after removing 50 from the above tree.

D. (5 pts) Assume that you are required to use the *insert* algorithm that we discussed in class to insert the value "77" to the AVL tree that you got from question C. Draw the AVL tree after inserting 77 to the AVL tree that you got from question C.

Q5. (10 pts) [**Recursive thinking, binary search**] Given the following binary search function,

```
private static int binarySearch (int[]array, int idxs, int idxe, int searche){
    System.out.println("["+idxs+","+idxe+"]");
    if(idxe<idxs) return (-1);
    int idx_middle = (idxe+idxs)/2;

    if(array[idx_middle]==searche)
            return idx_middle;
    else if(searche<array[idx_middle])
            return binarySearch(array,idxs,idx_middle-1,searche);
    else
            return binarySearch(array,idx_middle+1,idxe,searche);
}
```

What is the complexity of the **binarySearch method** in Big-O ____________________________

define n= ________________

What is the output after running the following 3 lines of code?

```
int[] A = {1, 3, 7, 11, 14};
int resultPos = binarySearch(A, 0,A.length-1,3); //note: this function prints information
System.out.println("="+resultPos);
```

Result:

Q6. (10 pts) **[Heap]** Implement a max heap class utilizing ArrayList to hold the elements. Finish the *reheapUpward* function. You can add other facilitating functions when necessary. What is the worst-case complexity of the **add method** in Big-O _______________

define n= ____

```
public class HeapArrayList {
        private ArrayList<Integer> elements;
        public HeapArrayList()          {elements=new ArrayList<Integer>();}

        public void add (int e){
                elements.add(e);
                reheapUpward(elements.size()-1);
        }

        public void reheapUpward(int pos){











        }

        //Other facilitating functions when necessary






}
```

Q7. (10 pts) [**Open-address hashing**] You are given the code for *Table* class. After running the main method, you are required to fill in the following space.
data[____________] = “obj1”, which has hash value ________________
data[____________] = “obj10”, which has hash value ________________
data[____________] = “obj20”, which has hash value ________________
What is the worst-case complexity of the **put method** in Big-O ________________
define n= ____

```
public class Table {
    private int num = 0;
    private Object[] keys = new Object[10];
    private Object[] data = new Object[10];
    private boolean[] used = new boolean[10];

    public Table()                  {for(int i=0;i<10;i++) used[i]=false;}
    private int hash(Object key){ return Math.abs(key.hashCode())%data.length; }

    public void put(Object  key, Object obj) throws Exception
    {
        if(num==data.length) throw new Exception("Table is full");
        int idx = findIndex(key);
        if(idx!=-1){data[idx] = obj;}
        else{
                idx = hash(key);
                while(used[idx]) idx = ((idx+1)==data.length)?0:(idx+1);
                keys[idx] = key;  data[idx] = obj; used[idx] = true;
                num++;
        }
    }

    private int findIndex (Object key) {
        int idx = hash(key);
        int count = 0;
        while(used[idx] & count<data.length){
                if(key.equals(keys[idx])) return idx;
                else idx = ((idx+1)==data.length)?0:(idx+1);
                count ++;
        }
        return -1;
    }

    public static void main(String[] args) throws Exception {
        Table tb = new Table();
        tb.put(1, "obj1");
        tb.put(10, "obj10");
        tb.put(20, "obj20");
    }
}
```

========================== END ==============================