SOLUTIONS

1. (6 pts) What is the output to the screen? (Note this is the same as the binary search algorithm discussed in class, with the addition of the println statement in the first line of the binarySearch method.)

```java
public class Problem1{
    public static void main(String[] args){
        int[] ar = {1,3,10,16,18,25};
        search(ar,11);
    }
    public static int search(int[] a, int val){
        return binarySearch(a,val,0,a.length-1);
    }
    public static int binarySearch(int[] a, int val, int min, int max){
        System.out.println("binarySearch: min = "+min+", max = "+max+".");
        if (max<min) // a base case
            return -1; // signifies not found
        else{
            int mid = (min+max)/2;
            if (a[mid] == val)
                return mid;
            else if (a[mid] > val) // left half
                return binarySearch(a,val,min,mid-1);
            else // right half
                return binarySearch(a,val,mid+1,max);
        }
    }
}
```

Solution:

binarySearch: min = 0, max = 5.
binarySearch: min = 3, max = 5.
binarySearch: min = 3, max = 3.
binarySearch: min = 3, max = 2.

2. (10 pts) Recall the interface GeneralFunction discussed in class:

```java
public interface GeneralFunction{
    public double functionValue (double x);
}
```
Write a class `testFunction` which implements the `GeneralFunction` interface. A `testFunction` object should represent the function

\[ e^{2x} - x^6. \]

Recall the exponential function \( e^x \) is given in Java by `Math.exp(x)`, while the power \( x^n \) is given by `Math.pow(x,n)`.

**Solution:**

```java
public class testFunction implements GeneralFunction{
    public double functionValue (double x){
        return Math.exp(2*x) - Math.pow(x,6);
    }
}
```

3. (6 pts) What is the return value \( f(5) \)? Show your work.

```java
public static int f(int n){
    if (n<=0) return 1;
    else return f(n-1) + f(n-3);
}
```

**Solution:** Compute \( f(5) = f(4) + f(2) = [f(3) + f(1)] + [f(1) + f(-1)] = f(3) + 2 \cdot f(1) + 1 = [f(2) + f(0)] + 2[f(0) + f(-2)] + 1 = f(2) + 1 + 2(1 + 1) + 1 = f(1) + f(-1) + 6 = f(1) + 1 + 6 = [f(0) + f(-2)] + 7 = 1 + 1 + 7 = 9. \)

4. Recall the `ListInterface` interface as discussed in class (and recall that we assume list indices go from 0 to \( \text{size}-1 \), like array indices).

```java
public interface ListInterface{
    // constructor cannot be part of an interface
    public boolean isEmpty ();
    public int size ();
    public void add (int index, Object item);
    public Object get (int index);
    public void remove (int index);
    public void removeAll();
}
```

(a) (6 pts) Assume `List` is a class which implements `ListInterface`. What is the output to the screen of the following code?

```java
ListInterface list = new List();
list.add(0,"carrot");
list.add(0,"peach");
list.add(0,"watermelon");
```
list.add(2,"cabbage");
list.remove(1);
String s = "apple";
list.add(2,s);
list.add(1,"pine"+s);
for (int i=0; i<list.size(); i++)
    System.out.println(list.get(i));

Solution:

watermelon
pineapple
cabbage
apple
carrot

(b) (10 pts) Using only the methods of the ListInterface, write a method

public static void reverseList(ListInterface x)

which reverses the order of the elements in x. You may need to use a temporary variable of class Object.

Solution:

public static void reverseList(ListInterface x){
    int s = x.size();
    for (int i=0; i<=(s-1)/2; i++){
        Object temp1 = x.get(i);
        Object temp2 = x.get(s-i-1);
        x.remove(i);
        x.add(i,temp2);
        x.remove(s-i-1);
        x.add(s-i-1,temp1);
    }
}

(c) (10 pts) Recall the class ListArrayBased, which implements ListInterface, and whose private data are

private static final int MAX_LIST = 50;
private Object items[]; // array of list items
private int numItems; // number of items in the list

Write the code for the method

public void remove (int index)

Solution:
public void remove (int index){
    if (index>=0 && index<numItems){
        for (int i=index; i<numItems-1; i++)
            items[i] = items[i+1];
        numItems--;
    }
}

5. (10 pts) Consider a linked list (such as ListReferenceBased discussed in class), which has private data

    private Node head;
    private int numItems;

Write a code fragment (inside a method of the linked list class) to insert item at the beginning of the list. You should use the public methods of the Node class:

public class Node{
    public Node(Object newItem)
    public Node(Object newItem, Node nextNode)
    public void setItem(Object newItem)
    public Object getItem()
    public void setNext (Node nextNode)
    public Node getNext()
}

Solution:

Node newNode = new Node(item,head);
head = newNode;
umItems--;

6. (10 pts) Write a recursive method

    public static double power (double x, int n)

whose return value is the power $x^n$, and which implements the following recursive algorithm:

$$
x^n = \begin{cases} 
1 & \text{if } n = 0 \\
(x^{n/2})^2 & \text{if } n > 0 \text{ and } n \text{ is even} \\
x \cdot (x^{n/2})^2 & \text{if } n > 0 \text{ and } n \text{ is odd} 
\end{cases}
$$

You may assume $n \geq 0$. Note the $n/2$ in the last line refers to integer division, in which any fractional part is dropped.

Solution:
public static double power (double x, int n){
    if (n==0)
        return 1;
    else if (n%2 == 0){
        double y = power(x,n/2);
        return y*y;
    }
    else{ // n is odd
        double y = power(x,n/2);
        return x*y*y;
    }
}