

1. In the space below, write Java code for a **recursive binary search method** in a class named SmartArray which searches the (sorted) array for a particular value. Your procedure receives three parameters; an object with a compareTo() method and beginning and ending subscripts of the portion of the array to be searched. The method returns a boolean (true/false) value.

```
public class SmartArray {
    public Comparable[] array;
    public int size; // number of elements in the array

    // other methods for this class ...

    public boolean binarySearch(Comparable x, int first, int last) {

}
}
```

2. In the space below, write Java code for a **recursive procedure** which computes the sum of the squares of the first N integers ( $N \geq 0$ ) and returns this value.

```
public int sumOfSquares (int N) {

}
}
```

3. The BinaryTree class has a data member named root which points to the root of the tree

```
private TreeNode root;
```

Assume the TreeNode class is defined as follows (with public data members, for convenience):

```
public class TreeNode {
    public TreeNode left, right;
    public Object data;

    public TreeNode(Object x) { //constructor
        data=x; left=null; right=null
    }
}
```

Write the Java code for doing an **inorder traversal** of a member of the BinaryTree class. The signature of the (recursive) method you need to implement is

```
private void inorder (Treenode t)
```

which places the nodes of the subtree rooted at t, in the order determined by an inorder traversal, into a queue named inOrderQueue.

```
private Queue inOrderQueue;

public void inorder() { // a top level method to get things started
    inOrderQueue = new Queue(); // create empty queue
    inorder(root); // traverse entire tree
}

private void inorder(TreeNode t) { // put t's nodes into the queue

}

}
```

4. A binary search tree is drawn below. Use this as your starting point for each of the following three parts.



(a) Draw the tree's structure after the node containing the value 91 has been deleted from the original tree above (using the standard deletion algorithm).

(b) Draw the tree's structure after the node containing the value 21 has been deleted from the original tree above (using the standard deletion algorithm).

(c) Draw the tree's structure after the node containing the value 47 has been deleted from the original tree above (using the standard deletion algorithm).





8. The data members of a weighted directed graph are

```
public String[] vertices;    // list of vertices
public int[][] edges;       // matrix of edge weights
public int numVertices;     // number of vertices
```

where a “vertex” is simply a string and an edge's weight is an integer. A weight of zero is a null value, corresponding to a “missing edge”.

In the spaces below, write Java code for `indexIs`, `addVertex` and `addEdge` methods for this class. Assume that no error conditions will be encountered.

```
public int indexIs(String v) { // find index of vertex v

}

}
```

```
public void addVertex(String newVertex) {

}

}
```

```
public void addEdge(String from, String to, int wt) {

}

}
```