

**QUESTION - 1**

```

import java.util.*;
public class UseOAHash {

    static class OAHash{
        int [] hashTable;
        int loadFactor;
        int currentCollisions = 0;
        boolean expandedOnce = false;
        public OAHash() {
            hashTable = new int[19];
            Arrays.fill(hashTable, -1);
            loadFactor = 0;
        }

        public int findHashValue(int k) {
            currentCollisions = 0;
            int functionOne = k % hashTable.length;
            if(hashTable[functionOne] != -1) {
                System.out.println("Collision Occurred at index "
+ functionOne);
                functionOne = handleCollision(functionOne, k, 1);
            }
            return functionOne;
        }

        public int handleCollision(int functionOne, int k, int
collisions) {
            int nextIndex = (functionOne + (13 - k %
13))%hashTable.length;
            if(hashTable[nextIndex] != -1) {
                System.out.println("Collision Occurred at index "
+ nextIndex);
                return handleCollision(nextIndex, k, collisions +
1);
            }
            currentCollisions = collisions;
            return nextIndex;
        }

        public void insert(int k) {
            loadFactor += 1;
            if(loadFactor >= (hashTable.length*0.75) &&
!expandedOnce) {

```

```

        System.out.println("Load Factor is about to
exceed 75%. Expanding table. ");
        expandedOnce = true;
        System.out.println("Contents of table before
expansion is as follows: ");
        contentsBeforeExpansion();
        increaseSize();
    }
    int indexToInsert = findHashValue(k);
    hashTable[indexToInsert] = k;
    System.out.println("Key " + k + " is inserted at index
" + indexToInsert);
    System.out.println ("Insertion of " + k + " resulted
in " + currentCollisions + " Collisions");
}

public void contentsBeforeExpansion() {
    System.out.print(hashTable[0]);
    for(int i = 1; i < hashTable.length; i++) {
        System.out.print(", " + hashTable[i]);
    }
    System.out.print(".\n");
}

public void increaseSize() {
    int[] temp = hashTable.clone();
    hashTable = new int[41];
    Arrays.fill(hashTable, -1);
    for(int i = 0; i < temp.length; i++) {
        if(temp[i] != -1)
            hashTable[temp[i] % 41] = temp[i];
    }
}

}

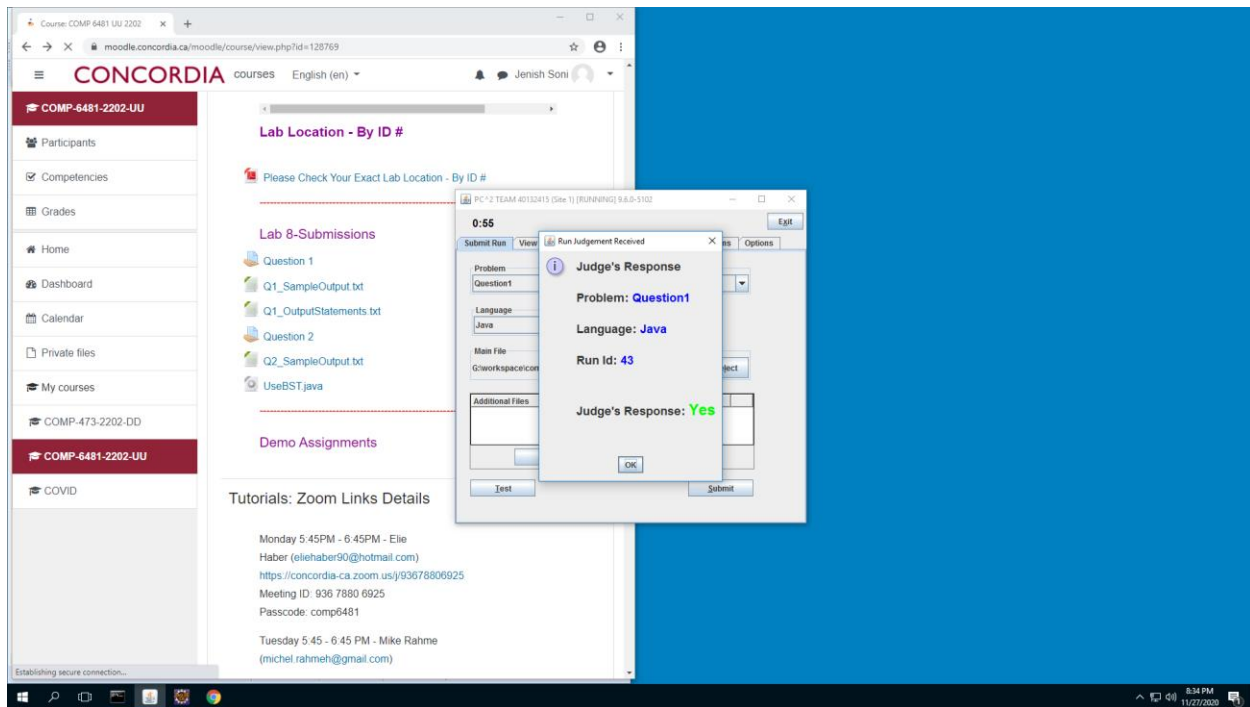
public static void main(String[] args) {
    System.out.println ("Enter a positive value that you want to
insert in the hash table; or -1 to terminate:");
    Scanner scan = new Scanner(System.in);
    OAHash hash = new OAHash();
    int numToInsert = scan.nextInt();
    while(numToInsert != -1) {
        hash.insert(numToInsert);
        numToInsert = scan.nextInt();
    }
    hash.contentsBeforeExpansion();
}

```

```

        scan.close();
    }
}

```



## QUESTION - 2

```

import java.util.*;
public class UseBST {

    public static void main(String[] args) {
        int y;
        BST b1 = new BST();
        System.out.println("Enter a value to add to the tree or -1
to terminate: ");
        int v;
        Scanner kb = new Scanner(System.in);
        v = kb.nextInt();
        while(v != -1)
        {
            b1.insert(v);
        }
    }
}

```

```

        System.out.println("Parent of " + v + " is " +
b1.parentOf(v));
        v = kb.nextInt();
    }
    y = b1.parentOf(126);
    if(y==-1)
        System.out.println("No Parent exists");
    else
        System.out.println("Parent of 126 is " + y);
    kb.close();
}
}

```

```

class BST{
    Node root;
    public BST() {
        @SuppressWarnings("unused")
        Node node = new Node();
    }

    public void insert(int value) {
        Node current = root;
        if(root == null) {
            root = new Node(value);
            return;
        }
        while(current != null) {
            if(current.value > value) {
                if(current.left == null) {
                    Node newNode = new Node(value);
                    current.left = newNode;
                    break;
                }
                else {
                    current = current.left;
                }
            }
            else {
                if(current.right == null) {
                    Node newNode = new Node(value);
                    current.right = newNode;
                    break;
                }
                else {
                    current = current.right;
                }
            }
        }
    }
}

```

```

    }
    }
}

public int parentOf(int value) {
    Node parentNode = null;
    Node current = root;
    if(current != null && value == current.value) return -1;
    while(current != null) {
        if(current.value > value) {
            parentNode = current;
            current = current.left;
        }
        else if(current.value < value){
            parentNode = current;
            current = current.right;
        }
        else {
            return parentNode.value;
        }
    }
    return -1;
}

class Node{
    Node left;
    Node right;
    public int value;

    public Node(int v) {
        value = v;
        left = right = null;
    }

    public Node() {
        value = 0;
        left = right = null;
    }
}
}

```

The screenshot shows the Eclipse IDE with a Java project named 'comp6481'. The main file 'UseBST.java' is open, displaying a program that implements a Binary Search Tree (BST). The program includes a 'main' method that takes command-line arguments and a 'BST' class with methods for inserting nodes and finding parent nodes. The console output shows the program's execution, including prompts for values to add to the tree and the resulting parent values for specific nodes.

```
1 import java.util.*;
2 public class UseBST {
3
4     public static void main(String[] args) {
5         int y;
6         BST b1 = new BST();
7         System.out.println("Enter a value to add to the tree or -1 to terminate: ");
8         int v;
9         Scanner kb = new Scanner(System.in);
10        v = kb.nextInt();
11        while(v != -1)
12        {
13            b1.insert(v);
14            System.out.println("Parent of " + v + " is ");
15            v = kb.nextInt();
16        }
17        y = b1.parentOf(126);
18        if(y == -1)
19            System.out.println("No Parent exists");
20        else
21            System.out.println("Parent of " + y + " is ");
22        kb.close();
23    }
24
25    class BST {
26        Node root;
27        public BST() {
28            @SuppressWarnings("unused")
29            Node node = new Node();
30        }
31
32        public void insert(int value) {
33            Node current = root;
34            if(root == null) {
35                root = new Node(value);
36                return;
37            }
38            while(current != null) {
39                if(current.value > value) {
40                    if(current.left == null) {
41                        Node newNode = new Node(value);
42                        current.left = newNode;
43                        break;
44                    }
45                }
46                else {
47                    current = current.right;
48                }
49            }
50        }
51    }
52
53    Node parentOf(int value) {
54        Node current = root;
55        if(current == null) return -1;
56        while(current != null) {
57            if(current.value == value) return current;
58            if(current.value > value) current = current.left;
59            else current = current.right;
60        }
61        return -1;
62    }
63
64    Node findParent(int value) {
65        Node current = root;
66        if(current == null) return -1;
67        while(current != null) {
68            if(current.value == value) return current;
69            if(current.value > value) current = current.left;
70            else current = current.right;
71        }
72        return -1;
73    }
74}
```

The console output shows the program's execution, including prompts for values to add to the tree and the resulting parent values for specific nodes:

```
Enter a value to add to the tree or -1 to terminate:
20
Parent of 20 is -1
32
Parent of 32 is 20
5
Parent of 5 is 20
is 5
is 5
is 32
is 45
is 45
is 2
is 15
is 9
is 9
is 52
is 67
is 67
Parent of 63 is 59
-1
No Parent exists
```

A 'Judge's Response' dialog box is overlaid on the console output, showing the following information:

- Problem: Question2
- Language: Java
- Run Id: 76
- Judge's Response: Yes