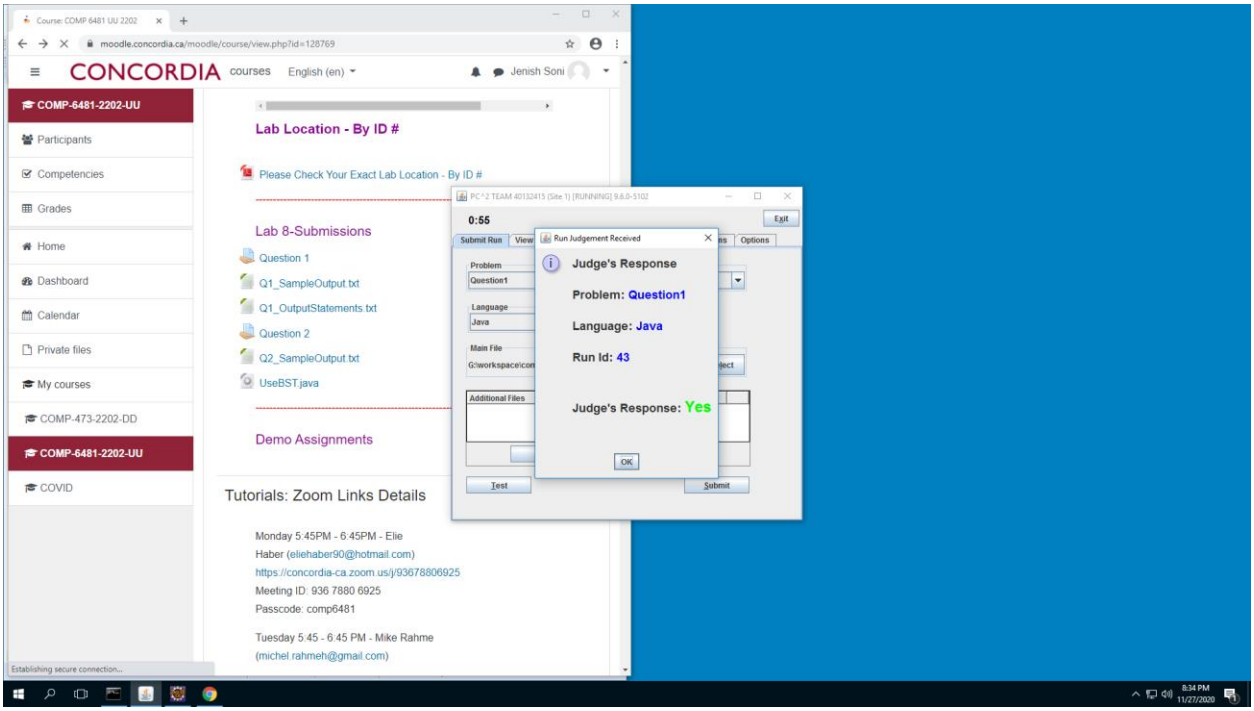
QUESTION - 1

```
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();
      }

}
```

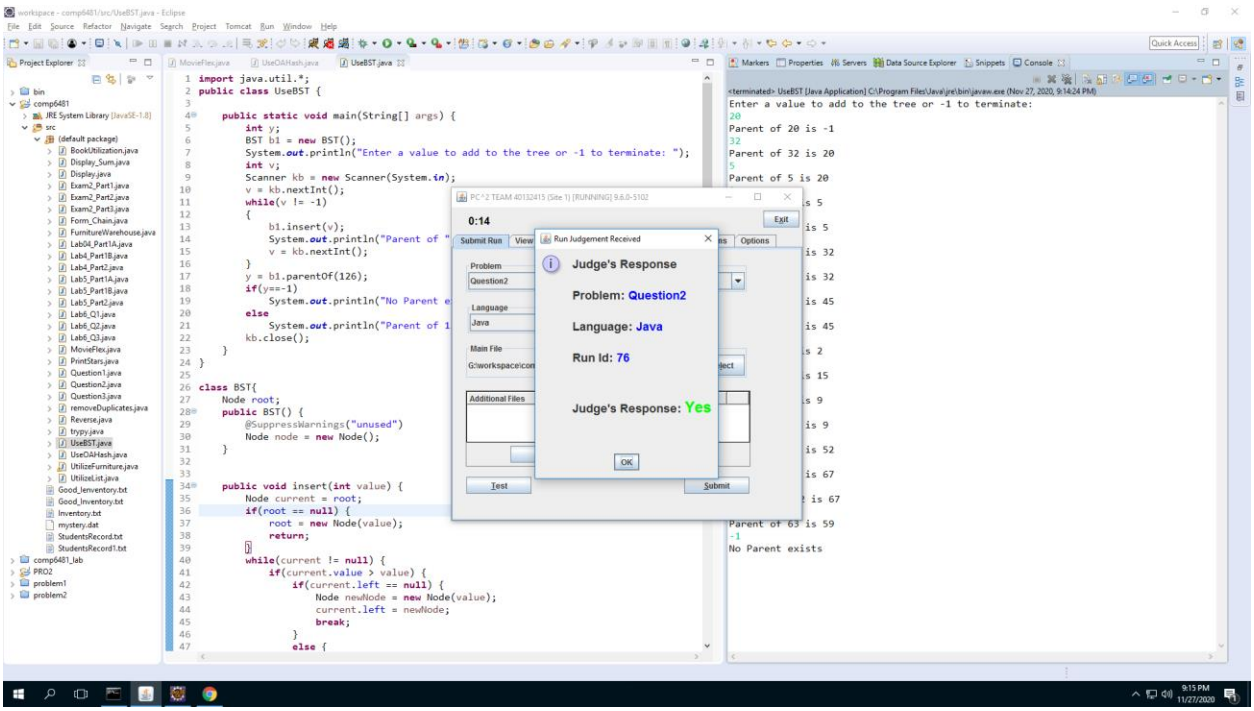


QUESTION - 2

```
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;
                }
            }
}
```


Enter a value to add to the tree or -1 to terminate:
Parent of 20 is -1
Parent of 32 is 20
Parent of 5 is 20
Parent of 63 is 59
No Parent exists
Judge's Response
Problem: Question2
Language: Java
Run Id: 76
Judge's Response: Yes
OK
Exit
Submit Run
Test
Submit