FULL STACK DEVELOPMENT – WORKSHEET – B

Ques1. What is the output for the below code ?

```
public class A {
public A(int i){
System.out.println(i);
}
}
1. public class B extends A{
2. public B(){
3. super(6);
4. this();
5. }
6. }
public class Test{
public static void main (String[] args){
B b = new B();
}
}
```

Ans: (D) -Compilation fails due to an error on lines 4

Ques2. Write a Java program to check if a vowel is present in a string.(like Hello-true
num -folse)

Ans:

```
public class VowelCheck {

    public static boolean containsVowel(String str) {
            // Convert the string to lower case to make the check case-insensitive
        str = str.toLowerCase();
            // Define a string containing all vowels
        String vowels = "aeiou";
            // Iterate through each character in the string
        for (int i = 0; i < str.length(); i++) {
            // Check if the current character is a vowel
            if (vowels.indexOf(str.charAt(i)) != -1) {
                return true;
            }
        }
        // If no vowel is found, return false
        return false;
    }

    public static void main(String[] args) {
        // Test cases
        String test1 = "Hello";
        String test2 = "num";

        System.out.println(test1 + " - " + containsVowel(test1)); // Output: Hello - true
        System.out.println(test2 + " - " + containsVowel(test2)); // Output: num - false
    }
}
```

Ques 3. Write a java program to Remove Duplicates elements from Array List.

Ans:

```
import java.util.ArrayList;
import java.util.HashSet;

public class RemoveDuplicates {

    public static <T> ArrayList<T> removeDuplicates(ArrayList<T> list) {
        // Create a HashSet to store unique elements
        HashSet<T> set = new HashSet<>(list);
        // Create a new ArrayList from the HashSet
        return new ArrayList<>(set);
    }

    public static void main(String[] args) {
        // Create an ArrayList with duplicate elements
        ArrayList<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(2);
        list.add(3);
        list.add(4);
        list.add(4);
        list.add(5);

        System.out.println("Original ArrayList: " + list);

        // Remove duplicates
        ArrayList<Integer> uniqueList = removeDuplicates(list);

        System.out.println("ArrayList after removing duplicates: " + uniqueList);
    }
}
```

Ques 4. Write a java Program to Union and Intersection of two Linked List

Ans:

```
import java.util.HashSet;

class Node {
    int data;
    Node next;

    public Node(int data) {
        this.data = data;
        this.next = null;
    }
}

class LinkedList {
    Node head;

    public void add(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
        } else {
            Node temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newNode;
        }
    }

    public void printList() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }

    public static LinkedList union(LinkedList list1, LinkedList list2) {
        HashSet<Integer> set = new HashSet<>();
        LinkedList result = new LinkedList();

        Node temp = list1.head;
        while (temp != null) {
            set.add(temp.data);
```

```
            result.add(temp.data);
            temp = temp.next;
        }

        temp = list2.head;
        while (temp != null) {
            if (!set.contains(temp.data)) {
                result.add(temp.data);
                set.add(temp.data);
            }
            temp = temp.next;
        }

        return result;
    }

    public static LinkedList intersection(LinkedList list1, LinkedList list2) {
        HashSet<Integer> set1 = new HashSet<>();
        HashSet<Integer> set2 = new HashSet<>();
        LinkedList result = new LinkedList();

        Node temp = list1.head;
        while (temp != null) {
            set1.add(temp.data);
            temp = temp.next;
        }

        temp = list2.head;
        while (temp != null) {
            if (set1.contains(temp.data)) {
                set2.add(temp.data);
            }
            temp = temp.next;
        }

        for (int data : set2) {
            result.add(data);
        }

        return result;
    }
}

public class Main {
    public static void main(String[] args) {
        LinkedList list1 = new LinkedList();
```

```
        LinkedList list2 = new LinkedList();

        list1.add(10);
        list1.add(15);
        list1.add(20);
        list1.add(25);

        list2.add(15);
        list2.add(25);
        list2.add(30);
        list2.add(35);

        System.out.print("List 1: ");
        list1.printList();

        System.out.print("List 2: ");
        list2.printList();

        LinkedList unionList = LinkedList.union(list1, list2);
        System.out.print("Union of List 1 and List 2: ");
        unionList.printList();

        LinkedList intersectionList = LinkedList.intersection(list1, list2);
        System.out.print("Intersection of List 1 and List 2: ");
        intersectionList.printList();
    }
}
```

Ques 5. Write a java Program to Sum of middle row and column in Matrix

Ans:

```
import java.util.Scanner;

public class MatrixMiddleSum {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Ask the user for the dimensions of the matrix
        System.out.print("Enter the number of rows in the matrix: ");
        int rows = scanner.nextInt();
        System.out.print("Enter the number of columns in the matrix: ");
        int cols = scanner.nextInt();

        // Initialize the matrix
        int[][] matrix = new int[rows][cols];

        // Populate the matrix
        System.out.println("Enter the elements of the matrix:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }

        // Identify the middle row and middle column
        int middleRow = rows / 2;
        int middleColumn = cols / 2;

        // Calculate the sum of the middle row
        int middleRowSum = 0;
        for (int j = 0; j < cols; j++) {
            middleRowSum += matrix[middleRow][j];
        }

        // Calculate the sum of the middle column
        int middleColumnSum = 0;
        for (int i = 0; i < rows; i++) {
            middleColumnSum += matrix[i][middleColumn];
        }

        // Print the results
        System.out.println("Sum of the middle row: " + middleRowSum);
        System.out.println("Sum of the middle column: " + middleColumnSum);

        // Close the scanner
```

```
    scanner.close();
    }
}
```

Ques 6. Write a java Program Merge two sorted linked lists

Ans.

```
class ListNode {
    int val;
    ListNode next;

    ListNode(int val) {
        this.val = val;
        this.next = null;
    }
}

public class MergeSortedLinkedLists {

    public static ListNode mergeTwoLists(ListNode l1, ListNode l2) {
        // Create a dummy node to serve as the start of the merged list
        ListNode dummy = new ListNode(0);
        ListNode current = dummy;

        // Traverse both lists
        while (l1 != null && l2 != null) {
            // Compare the values and add the smaller one to the merged list
            if (l1.val <= l2.val) {
                current.next = l1;
                l1 = l1.next;
            } else {
                current.next = l2;
                l2 = l2.next;
            }
            current = current.next;
        }

        // If one of the lists is not empty, append the rest to the merged list
        if (l1 != null) {
            current.next = l1;
        } else {
            current.next = l2;
        }

        // The merged list starts from the next of the dummy node
```

```
        return dummy.next;
    }

    public static void printList(ListNode node) {
        while (node != null) {
            System.out.print(node.val + " ");
            node = node.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
        // Create first sorted linked list: 1 -> 3 -> 5
        ListNode l1 = new ListNode(1);
        l1.next = new ListNode(3);
        l1.next.next = new ListNode(5);

        // Create second sorted linked list: 2 -> 4 -> 6
        ListNode l2 = new ListNode(2);
        l2.next = new ListNode(4);
        l2.next.next = new ListNode(6);

        // Print initial lists
        System.out.println("List 1:");
        printList(l1);
        System.out.println("List 2:");
        printList(l2);

        // Merge the two sorted lists
        ListNode mergedList = mergeTwoLists(l1, l2);

        // Print the merged sorted list
        System.out.println("Merged List:");
        printList(mergedList);
    }
}
```

Ques 7. Write a java Program to Print Bottom View of Binary Tree

Ans:

```
import java.util.*;

class TreeNode {
    int val;
    TreeNode left, right;

    TreeNode(int val) {
        this.val = val;
        left = right = null;
    }
}

public class BinaryTree {

    TreeNode root;

    public BinaryTree() {
        root = null;
    }

    // Method to print the bottom view of the binary tree
    public void printBottomView() {
        if (root == null) {
            return;
        }

        // TreeMap to store the horizontal distance and the bottom-most node's value
at that distance
        TreeMap<Integer, Integer> bottomViewMap = new TreeMap<>();

        // Queue for level order traversal. Stores pairs of node and its horizontal
distance
        Queue<Pair> queue = new LinkedList<>();

        // Start with the root node at horizontal distance 0
        queue.add(new Pair(root, 0));

        while (!queue.isEmpty()) {
            Pair temp = queue.poll();
            TreeNode node = temp.node;
            int hd = temp.hd;
```

```
        // Overwrite the map entry at the horizontal distance with the current node's
value
        bottomViewMap.put(hd, node.val);

        // If the node has a left child, add it to the queue with horizontal distance hd-
1
        if (node.left != null) {
            queue.add(new Pair(node.left, hd - 1));
        }

        // If the node has a right child, add it to the queue with horizontal distance
hd+1
        if (node.right != null) {
            queue.add(new Pair(node.right, hd + 1));
        }
    }

    // Print the bottom view
    for (Map.Entry<Integer, Integer> entry : bottomViewMap.entrySet()) {
        System.out.print(entry.getValue() + " ");
    }
    System.out.println();
  }

  // Helper class to store the node and its horizontal distance
  class Pair {
    TreeNode node;
    int hd;

    Pair(TreeNode node, int hd) {
        this.node = node;
        this.hd = hd;
    }
  }

  public static void main(String[] args) {
    BinaryTree tree = new BinaryTree();

    // Construct the binary tree
    tree.root = new TreeNode(20);
    tree.root.left = new TreeNode(8);
    tree.root.right = new TreeNode(22);
    tree.root.left.left = new TreeNode(5);
    tree.root.left.right = new TreeNode(3);
    tree.root.right.left = new TreeNode(4);
    tree.root.right.right = new TreeNode(25);
```

```
        tree.root.left.right.left = new TreeNode(10);
        tree.root.left.right.right = new TreeNode(14);

        System.out.println("Bottom view of the binary tree:");
        tree.printBottomView();
    }
}
```

Ques 8. Write a java Program to Convert a Binary Tree into its Mirror Tree

Ans:

```
class TreeNode {
    int val;
    TreeNode left, right;

    TreeNode(int val) {
        this.val = val;
        left = right = null;
    }
}

public class BinaryTree {

    TreeNode root;

    public BinaryTree() {
        root = null;
    }

    // Function to convert the binary tree to its mirror
    public void convertToMirror(TreeNode node) {
        if (node == null) {
            return;
        }

        // Swap left and right subtrees
        TreeNode temp = node.left;
        node.left = node.right;
        node.right = temp;

        // Recursively convert left and right subtrees
```

```
        convertToMirror(node.left);
        convertToMirror(node.right);
    }

    // Function to print the inorder traversal of the tree (for verification)
    public void inorderTraversal(TreeNode node) {
        if (node == null) {
            return;
        }
        inorderTraversal(node.left);
        System.out.print(node.val + " ");
        inorderTraversal(node.right);
    }

    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();

        // Construct the binary tree
        tree.root = new TreeNode(1);
        tree.root.left = new TreeNode(2);
        tree.root.right = new TreeNode(3);
        tree.root.left.left = new TreeNode(4);
        tree.root.left.right = new TreeNode(5);

        System.out.println("Original tree (inorder traversal):");
        tree.inorderTraversal(tree.root);
        System.out.println();

        // Convert the tree to its mirror
        tree.convertToMirror(tree.root);

        System.out.println("Mirror tree (inorder traversal after conversion):");
        tree.inorderTraversal(tree.root);
        System.out.println();
    }
}
```

Ques 9. Write a java Program to Determine if given Two Trees are Identical or not

Ans:

```
class TreeNode {
    int val;
    TreeNode left, right;

    TreeNode(int val) {
        this.val = val;
        left = right = null;
    }
}

public class BinaryTree {

    TreeNode root1, root2;

    public BinaryTree() {
        root1 = root2 = null;
    }

    // Function to check if two binary trees are identical
    public boolean checkIdentical(TreeNode node1, TreeNode node2) {
        // Base cases:
        // 1. Both nodes are null (identical leaf nodes)
        if (node1 == null && node2 == null) {
            return true;
        }

        // 2. One of the nodes is null and the other is not (different structure)
        if (node1 == null || node2 == null) {
            return false;
        }

        // 3. Both nodes have different values
        if (node1.val != node2.val) {
            return false;
        }

        // Recursively check left and right subtrees
        return checkIdentical(node1.left, node2.left) && checkIdentical(node1.right,
node2.right);
    }

    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();
```

```
        // Construct the first binary tree
        tree.root1 = new TreeNode(1);
        tree.root1.left = new TreeNode(2);
        tree.root1.right = new TreeNode(3);
        tree.root1.left.left = new TreeNode(4);
        tree.root1.left.right = new TreeNode(5);

        // Construct the second binary tree
        tree.root2 = new TreeNode(1);
        tree.root2.left = new TreeNode(2);
        tree.root2.right = new TreeNode(3);
        tree.root2.left.left = new TreeNode(4);
        tree.root2.left.right = new TreeNode(5);

        // Check if the two trees are identical
        if (tree.checkIdentical(tree.root1, tree.root2)) {
            System.out.println("The two binary trees are identical.");
        } else {
            System.out.println("The two binary trees are not identical.");
        }
    }
}
```

Ques 10. Write a java Program to find whether a no is power of two or not

Ans:

```
public class PowerOfTwo {

    // Function to check if a number is a power of two
    public static boolean isPowerOfTwo(int n) {
        // Handle edge case where n is non-positive
        if (n <= 0) {
            return false;
        }

        // Check if there is exactly one bit set in the binary representation of n
        return (n & (n - 1)) == 0;
    }

    public static void main(String[] args) {
        // Test cases
        int[] numbers = {1, 2, 3, 4, 8, 16, 32, 64, 128, 0, -1};

        for (int num : numbers) {
            System.out.println(num + " is power of two: " + isPowerOfTwo(num));
        }
    }
}
```