Assessment

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();
     }
}
```

A. 0 B. 6 C. Compilation fails due to an error on lines 3 D. Compilation fails due to an error on lines 4

Class A has a constructor that takes an int parameter and prints it.

Class B Extends class A.

Has a no-argument constructor that attempts to call super(6) to invoke the constructor of class A with the argument 6.

After super(6), it calls this(), which is supposed to call another constructor in the same class (B). However, the this() call here is incorrect because there is no constructor in class B that takes no arguments, and constructors cannot call themselves in this way, leading to a compile-time error.

So, the output is **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.

```
import java.util.Scanner;

Class Main{
public static boolean containsVowel(String input){

//convert the string to lowercase to handle case insensitive matching
String lowerCaseInput = input.toLowerCase();

//define a string containing all vowels

String vowels = “aeiou”;

// check if any of the characters in the input string is a vowel or not

for(char c : lowerCaseInput.toCharArray()){
if(vowels.indexOf(c) != -1){

return  true;

}

}

return false;

}

public static void main(String[] args){

Scanner sc =new Scanner(System.in);

String test = sc.next();

System.out.println(test + “ - “ + containsVowel(test));

}

}
```

Input : Hello

Output : Hello - true

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

```
import java.util.*;
```

```
public class RemoveDuplicates {
//<T> is a generic type parameter.
    public static <T> List<T> removeDuplicates(List<T> list) {
        // Create a new LinkedHashSet from the list to remove duplicates while maintaining order
        Set<T> set = new LinkedHashSet<>(list);
        // Convert the set back to a list
        return new ArrayList<>(set);
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
       // Create an ArrayList to store user input
        ArrayList<String> listWithDuplicates = new ArrayList<>();
        // Prompt user for input
        System.out.println("Enter the number of elements in the list:");
        int n = scanner.nextInt();
        scanner.nextLine();  // Consume the newline
        System.out.println("Enter the elements of the list:");
        for (int i = 0; i < n; i++) {
            String element = scanner.nextLine();
            listWithDuplicates.add(element);
        }
// Remove duplicates
       List<String> listWithoutDuplicates = removeDuplicates(listWithDuplicates);
        // Print the result
```

```
        System.out.println("Original list: " + listWithDuplicates);
        System.out.println("List after removing duplicates: " + listWithoutDuplicates);
    }
}
```

Input : Enter the number of elements in the list: 5

Enter the elements of the list:

apple

banana

apple

orange

banana

Output : Original list: [apple, banana, apple, orange, banana]

List after removing duplicates: [apple, banana, orange]

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

```
import java.util.*;
public class LinkedListOperations {
    // Method to find the union of two linked lists
    public static LinkedList<Integer> union(LinkedList<Integer> list1, LinkedList<Integer>
list2) {
        HashSet<Integer> set = new HashSet<>();
        LinkedList<Integer> unionList = new LinkedList<>();
        // Add all elements from list1 to the set
        set.addAll(list1);
        // Add all elements from list2 to the set
```

```
        set.addAll(list2);
        // Add all elements from the set to the unionList
        unionList.addAll(set);
        return unionList;
    }
    // Method to find the intersection of two linked lists
    public static LinkedList<Integer> intersection(LinkedList<Integer> list1,
LinkedList<Integer> list2) {
        HashSet<Integer> set1 = new HashSet<>(list1);
        LinkedList<Integer> intersectionList = new LinkedList<>();
        // Iterate through list2 and add common elements to the intersectionList
        for (Integer element : list2) {
            if (set1.contains(element)) {
                intersectionList.add(element);
            }
        }
        return intersectionList;
    }
    public static void main(String[] args) {
        // Creating first linked list
        LinkedList<Integer> list1 = new LinkedList<>();
        list1.add(1);
        list1.add(2);
        list1.add(3);
        list1.add(4);
```

```
        list1.add(5);
        // Creating second linked list
        LinkedList<Integer> list2 = new LinkedList<>();
        list2.add(3);
        list2.add(4);
        list2.add(5);
        list2.add(6);
        list2.add(7);
        // Finding the union of the two linked lists
        LinkedList<Integer> unionList = union(list1, list2);
        System.out.println("Union of the two linked lists: " + unionList);
        // Finding the intersection of the two linked lists
        LinkedList<Integer> intersectionList = intersection(list1, list2);
        System.out.println("Intersection of the two linked lists: " + intersectionList);
    }
}
```

Output : Union of the two linked lists: [1, 2, 3, 4, 5, 6, 7]

Intersection of the two linked lists: [3, 4, 5]

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

```
public class Main{
 // function to calculate the sum of the middle row of a matrix
  public static int sumOfMiddleRow(int [][] matrix, int n, int m){
    int totalSum =0; //variable to store the total sum value
```

```
    // Iterating over the middle column and picking the middle value
    for(int col = 0; col<m; col++){
        totalSum += matrix[n/2][col];
    }
    return totalSum;
  }
  // function to calculate the sum of the middle column of a matrix
  public static int sumOfMiddleColumn(int [][] matrix, int n, int m){
    int totalSum =0; //variable to store the total sum value
    // Iterating over all rows and picking the middle value
    for(int row = 0; row<n; row++){
        totalSum += matrix[row][m/2];
    }
    return totalSum;
  }
   public static void main(String[] args) {
    int n= 3; // number of rows
    int m = 3; // number of columns
    // Input
    int [][]matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    System.out.println("Sum of the middle row: " +
Integer.toString(sumOfMiddleRow(matrix,n,m)));
    System.out.println("Sum of the middle column: "+
Integer.toString(sumOfMiddleColumn(matrix,n,m)));
  }
```

```
}
```

Output : Sum of the middle row: 15

Sum of the middle column: 15

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

```
import java.util.*;
// create Node
class Node{
    int data;
    Node next;
    Node(int d) {
        data = d;
        next = null;
    }
}
class Main{
    Node head;
    // add to last method to add node at last
    public void addToTheLast(Node node)    {
        if (head == null)    head = node;
        else{
            Node temp = head;
            while (temp.next != null)
                temp = temp.next;
```

```
        temp.next = node;
    }
}
//print method to print the linkedList
void printList()   {
    Node temp = head;
    while (temp != null)
    {
        System.out.print(temp.data + " ");
        temp = temp.next;
    }
    System.out.println();
}
public static void main(String args[])  {
    Main head1 = new Main(); // list 1
    Main head2 = new Main(); // list 2
    // insert nodes in 1st list
    head1.addToTheLast(new Node(1));
    head1.addToTheLast(new Node(2));
    head1.addToTheLast(new Node(4));
    head1.addToTheLast(new Node(6));
    head1.addToTheLast(new Node(9));
    // insert nodes in 2nd list
    head2.addToTheLast(new Node(3));
```

```
        head2.addToTheLast(new Node(4));
        head2.addToTheLast(new Node(7));
        head2.addToTheLast(new Node(8));
        // call merge sorted list method
        head1.head = new Mergesortedlists().MergeSortedLists(head1.head, head2.head);
        // print output
        head1.printList();
        }
}
class Mergesortedlists {
    // method to merge two sorted lists
    public Node MergeSortedLists(Node head1, Node head2)  {
        //check head is null or not
        if(head1 == null) return head2;
        if(head2 == null) return head1;
        // recursive function after check which head data is small
        if(head1.data < head2.data)   {
            head1.next = MergeSortedLists(head1.next, head2);
            return head1;
        }
        else   {
            head2.next = MergeSortedLists(head1, head2.next);
            return head2;
        }
```

```
    }
}
```

Output : 1 2 3 4 4 6 7 8 9

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

```
import java.util.*;
class Main{
    //Create new nodes with the class Node
    static class Node{
        /* class Node contains:
         ->node's value
         ->horizontal distance (hd)
         ->left and right child node */
        int val; //node's value
        int hd;  //horizontal distance
        // Left and right child nodes
        Node left;
        Node right;
        //Constructor
        public Node(int val){
            this.val = val;
            this.hd = Integer.MAX_VALUE; //hd will be updated later
            this.left = null;
            this.right = null;
```

```
        }
    }
    static void bottomViewHelper(Node node, int height, int hd, TreeMap<Integer, int[]>
map){
        //A base case that returns when there's no node
        if(node == null) return;
        //If any node for the current horizontal distance already exists, and
        // the current node's height is greater than that of previous one
        //we'll update it with the current node's value and height
        if(map.containsKey(hd) == true){
            int[] existing = map.get(hd);
            //if height is greater
            if(existing[1] <= height){
                existing[0] = node.val; //putting new node's value
                existing[1] = height; //updating height
            }
            map.put(hd, existing); //update and add to the map
        }else{
            //when the horizontal distance is not present in the map,
            //we'll simply add the current one
            map.put(hd, new int[]{node.val, height}); //add to the map
        }
        //Recursively call left child node
        bottomViewHelper(node.left,  height+1, hd-1, map);
        //Recursively call right child node
```

```
    bottomViewHelper(node.right, height+1, hd+1, map);
}
static void bottomView(Node root){
    //TreeMap
    //Key -> horizontal distance,
    //value -> {node's value, height}
    TreeMap<Integer, int[]> map = new TreeMap<>();
    //Calling helper method
    bottomViewHelper(root, 0, 0, map);
    //Printing the bottom-most nodes of the binary tree
    for(int arrays[]: map.values()){
        System.out.print(arrays[0] + " ");
    }
}
public static void main(String[] args){
    //Create a binary tree
    Node root = new Node(3);
    root.left = new Node(6);
    root.right = new Node(12);
    root.left.left = new Node(5);
    root.left.right = new Node(11);
    root.right.left = new Node(7);
    root.right.right = new Node(8);
    root.left.right.left = new Node(20);
```

```
        root.left.right.right = new Node(17);
        System.out.print("The bottom view of the binary tree is: ");
        bottomView(root);
    }
}
```

Output : The bottom view of the binary tree is: 5 20 7 17 8

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

```
// Iterative Java program to convert a Binary.
// Tree to its mirror
import java.util.*
class Main {
        /* A binary tree node has data, pointer to left child and a pointer to right child */
        static class Node {
                int data;
                Node left;
                Node right;
        };
        /* Helper function that allocates a new node with the given data and null left and
right pointers. */
        static Node newNode(int data){
                Node node = new Node();
                node.data = data;
                node.left = node.right = null;
                return (node);
```

```
    }
    /* Change a tree so that the roles of the left and right pointers are swapped at every
node.
    So the tree...
            4
            / \
            2 5
            / \
          1 3
    is changed to...
            4
            / \
            5 2
                / \
            3 1
    */
    static void mirror(Node root){
            if (root == null)  return;
            Queue<Node> q = new LinkedList<>();
            q.add(root);
            // Do BFS. While doing BFS, keep swapping
            // left and right children
            while (q.size() > 0) {
                    // pop top node from queue
                    Node curr = q.peek();
```

```
                q.remove();
                // swap left child with right child
                Node temp = curr.left;
                curr.left = curr.right;
                curr.right = temp;
                // push left and right children
                if (curr.left != null)    q.add(curr.left);
                if (curr.right != null)  q.add(curr.right);
        }
}
/* Helper function to print Inorder traversal.*/
static void inOrder(Node node){
        if (node == null)       return;
        inOrder(node.left);
        System.out.print(node.data + " ");
        inOrder(node.right);
}
public static void main(String args[]){
        Node root = newNode(1);
        root.left = newNode(2);
        root.right = newNode(3);
        root.left.left = newNode(4);
        root.left.right = newNode(5);
        /* Print inorder traversal of the input tree */
```

```
        System.out.print("Inorder traversal of the"+ " constructed tree is \n");
        inOrder(root);
        /* Convert tree to its mirror */
        mirror(root);
        /* Print inorder traversal of the mirror tree */
        System.out.print("\nInorder traversal of the "+ "mirror tree is \n");
        inOrder(root);
    }
}
```

Output : Inorder traversal of the constructed tree is 4 2 5 1 3

Inorder traversal of the mirror tree is 3 1 5 2 4

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

```
import java.util.*;
public class Main {
    A binary tree node
   static class Node {
      int data;
      Node left, right;
      public Node(int data) { this.data = data; }
   }
   // Driver code
   public static void main(String[] args){
      Node root1 = new Node(1);
      root1.left = new Node(2);
```

```
    root1.right = new Node(3);
    root1.left.left = new Node(4);
    root1.left.right = new Node(5);

    Node root2 = new Node(1);
    root2.left = new Node(2);
    root2.right = new Node(3);
    root2.left.left = new Node(4);
    root2.left.right = new Node(5);
      // Function call
    if (isIdentical(root1, root2)) {
        System.out.println(
            "Both the trees are identical.");
    }
    else {
        System.out.println(
            "Given trees are not identical.");
    }
}
// Function to check if two trees are identical
static boolean isIdentical(Node root1, Node root2){
    // Create two arraylist to store traversals
    ArrayList<Integer> res1 = new ArrayList<Integer>();
    ArrayList<Integer> res2 = new ArrayList<Integer>();
```

```
    // check inOrder
    inOrder(root1, res1);
    inOrder(root2, res2);
    if (!res1.equals(res2))       return false;
    // clear previous result to reuse arraylist
    res1.clear();
    res2.clear();
    // check PreOrder
    preOrder(root1, res1);
    preOrder(root2, res2);
    if (!res1.equals(res2))      return false;
    // clear previous result to reuse arraylist
    res1.clear();
    res2.clear();
    // check PostOrder
    postOrder(root1, res1);
    postOrder(root2, res2);
    if (!res1.equals(res2))     return false;
    return true;
}
// Utility function to check inorder traversal
static void inOrder(Node root, ArrayList<Integer> sol){
    if (root == null)       return;
    inOrder(root.left, sol);
```

```
        sol.add(root.data);
        inOrder(root.right, sol);
    }
    // Utility function to check preorder traversal
    static void preOrder(Node root, ArrayList<Integer> sol){
        if (root == null)          return;
        sol.add(root.data);
        preOrder(root.left, sol);
        preOrder(root.right, sol);
    }
    // Utility function to check postorder traversal
    static void postOrder(Node root, ArrayList<Integer> sol){
        if (root == null)          return;
        postOrder(root.left, sol);
        postOrder(root.right, sol);
        sol.add(root.data);
    }
}
```

Output : Both the trees are identical

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

```
import java.*;
class Main {
    // Function to check if x is power of 2
```

```
    static boolean isPowerOfTwo(int n)
    {
        if (n == 0)
            return false;
// Math function to check power
        return (int)(Math.ceil((Math.log(n) / Math.log(2))))
            == (int)(Math.floor(
                ((Math.log(n) / Math.log(2)))));
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int x = sc.nextInt();
        // Function call
        if (isPowerOfTwo(x))
            System.out.println("Yes");
        else
            System.out.println("No");
    }
}
```

Input : 4

Output : Yes