JAVA LAB MST ASSIGNMENT

Q1. The next greater element…

```
import java.util.*;
public class NextGreaterElementFinder {
    public int[] nextGreaterElement(int[] nums1, int[] nums2) {
        Map<Integer, Integer> nextGreater = new HashMap<>();
        Stack<Integer> stack = new Stack<>();

        for (int num : nums2) {
            while (!stack.isEmpty() && num > stack.peek()) {
                nextGreater.put(stack.pop(), num);
            }
            stack.push(num);
        }

        while (!stack.isEmpty()) {
            nextGreater.put(stack.pop(), -1);
        }

        int[] result = new int[nums1.length];
        for (int i = 0; i < nums1.length; i++) {
            result[i] = nextGreater.get(nums1[i]);
        }

        return result;
    }

    public static void main(String[] args) {
        NextGreaterElementFinder finder = new NextGreaterElementFinder();
```

```
        int[] nums1 = {4, 1, 2};
        int[] nums2 = {1, 3, 4, 2};

        int[] result = finder.nextGreaterElement(nums1, nums2);
        System.out.println(Arrays.toString(result));
    }
}
```

Q2.Develop a program showing the concept of inheritance.

```
class Animal {
    String name;
    int age;

    void eat() {
        System.out.println(name + " is eating.");
    }

    void sleep() {
        System.out.println(name + " is sleeping.");
    }
}

class Dog extends Animal {
    String breed;

    void bark() {
        System.out.println(name + " is barking.");
    }

    void displayInfo() {
```

```
        System.out.println("Name: " + name);
        System.out.println("Age: " + age + " years");
        System.out.println("Breed: " + breed);
    }
}

public class InheritanceDemo {
    public static void main(String[] args) {
        Dog myDog = new Dog();
        myDog.name = "Buddy";
        myDog.age = 3;
        myDog.breed = "Golden Retriever";

        myDog.displayInfo();
        myDog.eat();
        myDog.sleep();
        myDog.bark();
    }
}
```

Q3.Program showing the use method overloading.

```
public class MathOperations {

    // Add two integers
    int operate(int a, int b) {
        return a + b;
    }

    // Multiply two doubles
    double operate(double a, double b) {
```

```
        return a * b;
    }

    // Subtract three integers
    int operate(int a, int b, int c) {
        return a - b - c;
    }

    // Concatenate two strings (overloading for non-math operation too)
    String operate(String a, String b) {
        return a + b;
    }

    public static void main(String[] args) {
        MathOperations mo = new MathOperations();

        System.out.println("Addition (int): " + mo.operate(5, 10));
        System.out.println("Multiplication (double): " + mo.operate(2.5, 4.0));
        System.out.println("Subtraction (int, 3 args): " + mo.operate(20, 5, 3));
        System.out.println("Concatenation (String): " + mo.operate("Hello, ", "World!"));
    }
}
```

Q4. Create a custom exception class…

```
import java.util.Scanner;

// Custom exception for invalid age
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
```

```
    }
}

// Custom exception for invalid name
class InvalidNameException extends Exception {
    public InvalidNameException(String message) {
        super(message);
    }
}

// Main class
public class MultipleCustomExceptionsDemo {

    // Method to validate name
    static void validateName(String name) throws InvalidNameException {
        if (name == null || name.trim().isEmpty() || name.length() < 2) {
            throw new InvalidNameException("Name must be at least 2 characters long and not
empty.");
        }
    }

    // Method to validate age
    static void validateAge(int age) throws InvalidAgeException {
        if (age < 18) {
            throw new InvalidAgeException("Age must be 18 or older to vote.");
        }
    }

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

```
        try {
            // Input name
            System.out.print("Enter your name: ");
            String name = scanner.nextLine();

            // Input age
            System.out.print("Enter your age: ");
            int age = scanner.nextInt();

            // Validation
            validateName(name);
            validateAge(age);

            System.out.println("\nWelcome " + name + "! You are eligible to vote.");

        } catch (InvalidNameException | InvalidAgeException e) {
            System.out.println("Validation Failed: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Unexpected error: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

Q5. Consider a function public string….

```
  public class PatternMatch {
  public static String matchFound(String input1, String input2) {
      String[] words = input2.split(":");
      StringBuilder output = new StringBuilder();
```

```
    for (String word : words) {
        if (isMatch(input1.toLowerCase(), word.toLowerCase())) {
            if (output.length() > 0) {
                output.append(":");
            }
            output.append(word.toUpperCase());
        }
    }

    return output.toString();
}

private static boolean isMatch(String pattern, String word) {
    if (pattern.length() != word.length()) return false;

    int diffCount = 0;

    for (int i = 0; i < pattern.length(); i++) {
        if (pattern.charAt(i) != '_') {
            if (pattern.charAt(i) != word.charAt(i)) {
                return false;
            }
        } else {
            diffCount++;
        }
    }

    return diffCount == 1;
}
```

```
    public static void main(String[] args) {
        String input1 = "c_t";
        String input2 = "cat:cut:cot:bat:car:cart";

        String output1 = matchFound(input1, input2);
        System.out.println("Matched Words: " + output1);
    }
}
```