Name -Jasmeet Kaur

UID -23BCS80047

Department: BE-CSE/IT 3rd Year

Subject: FOUNDATION OF CLOUD IOT EDGE ML LAB
Subject Code: 22CSP-367/22ITP-367
Semester: 6th
Batch: 2026

Lab Based Average Coding Problems

QUESTION 1.

The next greater element of some element x in an array is the first greater element that is to the right of x in the same array.

You are given two distinct 0-indexed integer arrays nums1 and nums2, where nums1 is a subset of nums2.

For each 0 <= i < nums1.length, find the index j such that nums1[i] == nums2[j] and determine the next greater element of nums2[j] in nums2. If there is no next greater element, then the answer for this query is -1.

Return an array ans of length nums1.length such that ans[i] is the next greater element as described above.

Hint:
Input: nums1 = [4,1,2], nums2 = [1,3,4,2]
Output: [-1,3,-1]
Explanation: The next greater element for each value of nums1 is as follows:
- 4 is underlined in nums2 = [1,3,4,2]. There is no next greater element, so the answer is -1.
- 1 is underlined in nums2 = [1,3,4,2]. The next greater element is 3.
- 2 is underlined in nums2 = [1,3,4,2]. There is no next greater element, so the answer is -1.

Code:

```
import java.util.*;

public class Q1_NextGreaterElement {
    public static int[] nextGreaterElement(int[] nums1, int[] nums2) {
        Map<Integer, Integer> map = new HashMap<>();
        Stack<Integer> stack = new Stack<>();

        for (int num : nums2) {
            while (!stack.isEmpty() && stack.peek() < num) {
                map.put(stack.pop(), num);
            }
```

```
            stack.push(num);
        }
        int[] res = new int[nums1.length];

        for (int i = 0; i < nums1.length; i++) {
            res[i] = map.getOrDefault(nums1[i], -1);
        }
        return res;
    }

    public static void main(String[] args) {
        int[] nums1 = {4, 1, 2};
        int[] nums2 = {1, 3, 4, 2};
        System.out.println("Output: " + Arrays.toString(nextGreaterElement(nums1, nums2)));
    }
}
```

Output:

Output: [-1, 3, -1]

QUESTION 2.

Develop a Java program showcasing the concept of inheritance. Create a base class and a derived class with appropriate methods and fields.

Code:

```
class Animal {
    void sound() {
        System.out.println("Animal makes sound");
    }
}

class Dog extends Animal {
    void sound() {
        System.out.println("Dog barks");
    }
}

public class Q2_Inheritance {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.sound();
    }
}
```

Output:

Dog barks

QUESTION 3.

Encoding Three Strings: Anand was assigned the task of coming up with an encoding mechanism for any given three strings.He has come up with the following plan.

Step ONE :- Given any three strings,break each string into 3 parts each.
For example- if the three strings are below:
Input 1: “John”
Input 2: “Johny”
Input 3 : “Janardhan”

“John” should be split into”J”,”oh”,”n,” as the FRONT ,MIDDLE and END part repectivly.

“Johny” should be spilt into “jo”,” h”, “ny” as the FRONT ,MIDDLE and END respectively.

“Janardhan” should be split into “Jan” ,” ard” ,”han” as the FRONT ,MIDDLE and END part respectively.

i.e. If the no. of characters in the string are in multiples of 3 ,then each split –part will contain equal no of characters , as seen in the example of “Janadhan”.

If the no. of characters in the string are NOT in multiples of 3 ,and if there is one character more than multiple of 3, then the middle part will get the extra character ,as seen in the example of “john”.

If the no. of characters in the string are Not in multiples of 3 and if there are two characters more than multiple of 3, then the FRONT and END parts will get one extra character each, as seen in the example of “Johny”.

Step TWO : Concatenate (join) the FRONT ,MIDDLE and END parts of the string as per the below specified concatenation – rule to from three Output strings.

Output 1: FRONT part of input 1 + MIDDLE part of input 2 +END part of input 3

Output2:- MIDDLE part of input1+ END part of input2 + FRONT part of input3

Output3: END part of the input1+FRONT part of input2 +MIDDLE part of input3

For example , for the above example input strings:

Output1 = “J” +”h”+han”=”jhhan”

Output2 =”oh”+”ny”+”Jan”=”ohnyjan”

Output3=”n”+Jo”+ard”+=“njoard”

Step THREE :-

Process the resulting output strings based on the output-processing rule .After the above two steps, we will now have three output string. Further processing is required only for the third output string as per below rule-

“Toggle the case of each character in the string “ ,i.e. in the third output string, all lower-case characters should be made upper-case and vice versa.

For example , for the above example strings ,output3 is “nJoard”, so after applying the toggle rule. Output3 should become “NjOARD”.

Final Result – The three output strings after applying the above three steps i.e. for the above example .

Output1 =”Jnhan”

Output2=”ohnyJan’

Output3 = “NJOARD”

Help Anand to write a program that would do the above.

Code:

```
public class Q3_EncodeThreeStrings {
   public static String[] splitString(String str) {
      int len = str.length();
      int rem = len % 3;
      int part = len / 3;
      String front, middle, end;
```

```
        if (rem == 0) {
            front = str.substring(0, part);

            middle = str.substring(part, 2 * part);
            end = str.substring(2 * part);
        } else if (rem == 1) {
            front = str.substring(0, part);
            middle = str.substring(part, 2 * part + 1);
            end = str.substring(2 * part + 1);
        } else {
            front = str.substring(0, part + 1);
            middle = str.substring(part + 1, 2 * part + 1);
            end = str.substring(2 * part + 1);
        }

        return new String[]{front, middle, end};
    }

    public static void main(String[] args) {
        String s1 = "John", s2 = "Johny", s3 = "Janardhan";
        String[] p1 = splitString(s1);
        String[] p2 = splitString(s2);
        String[] p3 = splitString(s3);

        String o1 = p1[0] + p2[1] + p3[2];
        String o2 = p1[1] + p2[2] + p3[0];
        String o3 = p1[2] + p2[0] + p3[1];

        StringBuilder sb = new StringBuilder();
        for (char c : o3.toCharArray()) {
            sb.append(Character.isLowerCase(c) ? Character.toUpperCase(c) : Character.toLowerCase(c));
        }

        System.out.println("Output1: " + o1);
        System.out.println("Output2: " + o2);
        System.out.println("Output3: " + sb.toString());
    }
}
```

Output:

```
Output1: Jhhhan
Output2: ohnyJan
Output3: NjOARD
```

QUESTION 4.

Implement a Java program that uses method overloading to perform different mathematical operations.

Code:

```
public class Q4_MethodOverloading {
    int add(int a, int b) {
        return a + b;
    }

    double add(double a, double b) {
        return a + b;
    }
```

```
    String add(String a, String b) {
        return a + b;
    }

    public static void main(String[] args) {
        Q4_MethodOverloading obj = new Q4_MethodOverloading();
        System.out.println(obj.add(2, 3));
        System.out.println(obj.add(2.5, 3.5));
        System.out.println(obj.add("Hello ", "World"));
    }
}
```

Output:

5

6.0

Hello World

QUESTION 5.

Given a String (In Uppercase alphabets or Lowercase alphabets), new alphabets is to be appended with following rule:

(i) If alphabet is present in input string, use numeric value of that alphabet. E.g. a or A numeric value is 1 and so on. New alphabet to be appended between 2 alphabets :

(a) If (sum of numeric value of 2 alphabets) %26 is 0, then append 0. E.g. string is ay. Numeric value of a is 1, y is 25. Sum is 26. Remainder is 0, new string will be a0y.

(b) Otherwise (sum of numeric value of 2 alphabets) %26 numeric value alphabet is to appended. E.g. ac is string. Numeric value of a is 1, c is 3, sum is 4. Remainder with 26 is 4. Alphabet to be appended is d. output will be adc.

(ii) If digit is present, it will be same in output string. E.g. string is 12, output string is 12.

(iii) If only single alphabet is present, it will be same in output string. E.g. input string is 1a, output will be 1a.

(iv) If space is present, it will be same in output string. E.g. string is ac 12a, output will be adc 12a.

Constraint: Whether string alphabets are In Uppercase or Lowercase, appended alphabets must be in lower case. Output string must also be in lowercase.

Code:

```
public class Q5_AppendAlphabet {
    public static String process(String str) {
```

```
        StringBuilder result = new StringBuilder();
        str = str.toLowerCase();
        int i = 0;

        while (i < str.length()) {
            char ch1 = str.charAt(i);
            if (Character.isLetter(ch1) && i + 1 < str.length() && Character.isLetter(str.charAt(i + 1))) {
                char ch2 = str.charAt(i + 1);
                int sum = (ch1 - 'a' + 1) + (ch2 - 'a' + 1);
                if (sum % 26 == 0) {
                    result.append(ch1).append("0");
                } else {
                    char mid = (char) ((sum % 26 - 1) + 'a');
                    result.append(ch1).append(mid);
                }
                i++;
            } else {
                result.append(ch1);
            }
            i++;
        }
        return result.toString();
    }

    public static void main(String[] args) {
        String input = "ac 12a";
        System.out.println("Output: " + process(input));
    }
}
```

Output:

Output: adc 12a

QUESTION 6. Define an interface in Java and create a class that implements it, demonstrating the concept of abstraction.

Code:

```
interface Shape {
    void draw();
}

class Circle implements Shape {
    public void draw() {
        System.out.println("Drawing Circle");
    }
}

public class Q6_InterfaceExample {
    public static void main(String[] args) {
        Shape shape = new Circle();
        shape.draw();
    }
}
```

Output:

Drawing Circle

QUESTION 7.

String t is generated by random shuffling string s and then add one more letter at a random position.

Return the letter that was added to t.

Hint:

Input: s = "abcd", t = "abcde"

Output: "e"

Code:

```
public class Q7_FindAddedCharacter {
    public static char findTheDifference(String s, String t) {
        int sum = 0;
        for (char c : t.toCharArray()) sum += c;
        for (char c : s.toCharArray()) sum -= c;
        return (char) sum;
    }

    public static void main(String[] args) {
        System.out.println("Added character: " + findTheDifference("abcd", "abcde"));
    }
}
```

Output:

```
Added character: e
```

QUESTION 8. Create a custom exception class in Java. Write a program that throws this custom exception in a specific scenario.

Code:

```
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}

public class Q8_CustomException {
    public static void checkAge(int age) throws InvalidAgeException {
        if (age < 18)
            throw new InvalidAgeException("Age must be at least 18");
        System.out.println("Access granted");
    }

    public static void main(String[] args) {
        try {
            checkAge(16);
        } catch (InvalidAgeException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }
}
```

Output:

Exception caught: Age must be at least 18

QUESTION 9.

Consider a function public String matchFound(String input 1, String input 2), where

- input1 will contain only a single word with only 1 character replaces by an underscore '_'
- input2 will contain a series of words separated by colons and no space character in between
- input2 will not contain any other special character other than underscore and alphabetic characters.

The methods should return output in a String type variable "output1" which contains all the words from input2 separated by colon which matches with input 1. All words in output1 should be in uppercase.

Code:

```
public class Q9_MatchFound {
    public static String matchFound(String input1, String input2) {
        String[] words = input2.split(":");
        StringBuilder output = new StringBuilder();

        for (String word : words) {
            if (word.length() == input1.length()) {
                boolean match = true;
                for (int i = 0; i < word.length(); i++) {
                    if (input1.charAt(i) != '_' && input1.charAt(i) != word.charAt(i)) {
                        match = false;
                        break;
                    }
                }
                if (match) {
                    output.append(word.toUpperCase()).append(":");
                }
            }
        }

        if (output.length() > 0)
            output.setLength(output.length() - 1); // remove last colon
        return output.toString();
    }

    public static void main(String[] args) {
        String input1 = "a_k";
        String input2 = "ark:ask:ack:btk";
        System.out.println("Output: " + matchFound(input1, input2));
    }
}
```

Output: ARK:ASK:ACK

QUESTION 10.

Explain the difference between the throw and throws keywords in Java. Provide examples illustrating their usage.

Code:

```
public class Q10_ThrowVsThrows {
    static void validateAge(int age) throws ArithmeticException {
        if (age < 18)
            throw new ArithmeticException("Not valid age");
        else
            System.out.println("Welcome!");
    }

    public static void main(String[] args) {
```

```
        try {
            validateAge(15);
        } catch (ArithmeticException e) {


            System.out.println("Caught Exception: " + e.getMessage());
        }
    }
}
```

Output:

Caught Exception: Not valid age