Unit-1

Scenario Based Problems (Logical problem-solving skills)

- Improving logical problem-solving skills involves developing the ability to analyze situations, identify patterns, and come up with effective solutions.
- Solving logical problems in Java programming involves understanding the problem, breaking it down into manageable parts, and implementing a solution using Java's syntax and features.

Strategies to Improve Logical Problem-Solving Skills:

- Practice Regularly:
    - Puzzles and Games: Engage in activities like Sudoku, chess, and logic puzzles.
    - Brain Teasers: Solve riddles and brain teasers to stimulate your thinking.
- Study Logic:
    - Formal Logic: Learn about formal logic, including logical fallacies and syllogisms.
    - Mathematics: Study mathematics, especially areas like algebra and geometry, which require logical reasoning.
- Develop Analytical Thinking:
    - Break Down Problems: Practice breaking complex problems into smaller, manageable parts.
    - Question Assumptions: Challenge assumptions and consider alternative perspectives.
- Improve Critical Thinking:
    - Evaluate Arguments: Practice evaluating the strength of different arguments.
    - Decision Making: Practice making decisions based on logical analysis rather than emotions.
- Learn from Others:
    - Observe: Watch how others solve problems and learn from their approaches.
    - Seek Feedback: Get feedback on your problem-solving methods and learn from it.

Number Based Problems:

Question1:

You are working as a software engineer at a financial technology company. Your team is developing a feature for a financial analysis tool that processes a list of transaction amounts. For one of the analytical reports, you need to find the sum of the second last digits of these transaction amounts. This information is used to identify specific patterns in the transactions for further analysis.

Write a program in Java to accomplish this task. The program should take an array of transaction amounts as input and output the sum of the second last digits of these amounts.

Example:

Given the transaction amounts:

- **123**
- **4567**
- **89**
- **12**
- **5**
- **-246**

The second last digits are:

- **2 (from 123)**
- **6 (from 4567)**
- **8 (from 89)**
- **1 (from 12)**
- **0 (from 5, as it does not have a second last digit)**
- **4 (from -246)**

The sum of these second last digits is: 2 + 6 + 8 + 1 + 0 + 4 = 21

Write a Java program that implements this logic and calculates the sum for any given list of transaction amounts.

Solution:

```
public class SumSecondLastDigits {

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

```
        int[] numbers = {123, 4567, 89, 12, 5, -246}; // Sample array of numbers
        int sum = 0;

        for (int i = 0; i < numbers.length; i++) {
            int absNumber = Math.abs(numbers[i]); // Handle negative numbers
            int secondLastDigit;

            if (absNumber < 10) {
                secondLastDigit = 0; // If the number has less than 2 digits, set second last digit to 0
            } else {
                secondLastDigit = (absNumber / 10) % 10; // Extract the second last digit
            }

            sum += secondLastDigit;
        }

        System.out.println("Sum of second last digits: " + sum); // Output the sum
    }
}
```

Question2:

You are working as a software engineer for a cybersecurity company. Your team is developing a feature for a security application that analyzes phone numbers to detect potential fraud. One of the tasks involves identifying unique digits in a given phone number. This information is crucial for identifying patterns and anomalies in the phone numbers.

Write a program in Java to accomplish this task. The program should take a single phone number as input and output the unique digits present in that number.

Example:

Given the phone number:

- **8675389**

The unique digits are:

- **6, 7, 5, 3, 9**

Write a Java program that implements this logic and finds the unique digits for any given phone number.

Solution:

```
import java.util.*;

class HelloWorld {

   public static void main(String[] args) {

      int number;

     Scanner scanner = new Scanner(System.in);


      System.out.print("Enter a number: ");

      number = scanner.nextInt();


      // Array to store the frequency of each digit (0-9)

      int ar[]=new int[10];


      for(int i=0;i<10;i++)
```

```
        { ar[i]=0;}

        // Process each digit in the number using a while loop
        while (number > 0) {
            int digit = number % 10;
            ar[digit] = ar[digit]+1; // Mark this digit as present
            number /= 10;
        }

        // Output the unique digits
        System.out.print("Unique digits: ");
        for (int i = 0; i < 10; i++) {
            if (ar[i]==1) {
                System.out.print(i + " ");
            }
        }
    }
}
```

Question3:

Legend has it that the town's founder, a mathematical savant, engraved a secret number on a hidden stone tablet deep in the Numerica forest. This number, rumored to possess mystical properties, is known only to a select few. According to ancient texts, the number inscribed is 122312. As a participant in the Digit Dash, your task is to determine which digit appears most frequently in this sacred number.

Which digit holds the title of being the most frequent in this mystical sequence?

This question immerses participants in the town's folklore and challenges them to apply their analytical skills to uncover statistical insights within the digits of the secret number. It invites them to embark on a journey of discovery reminiscent of uncovering hidden truths in numerical data.

Write a Java Program to solve this.

Solution:

```
import java.util.*;

class HelloWorld {

   public static void main(String[] args) {

      int number;

     Scanner scanner = new Scanner(System.in);


      System.out.print("Enter a number: ");

      number = scanner.nextInt();


      // Array to store the frequency of each digit (0-9)

      int ar[]=new int[10];


      for(int i=0;i<10;i++)

      { ar[i]=0;}


      // Process each digit in the number using a while loop

      while (number > 0) {
```

```
            int digit = number % 10;
            ar[digit] = ar[digit]+1; // Mark this digit as present
            number /= 10;
        }


         int big=0;
        for (int i = 0; i < 10; i++) {
            if (ar[i]>big) {
                big=i;
            }
        }
     System.out.println(big);


    }
}
```

Question 4:

A stable number is a number in which the frequency of all digits is the same. Given this definition, determine if the number 123123 is stable or unstable. Explain your reasoning.
A number is considered stable if all its digits appear with the same frequency. Based on this, check if the number 778899 is stable or unstable. Given that a stable number has the same frequency for all its digits, verify if the number is stable or unstable by checking the frequency of each digit.

Ex:1

Input: 223113

Output: stable

Ex:2

Input: 2441

Output:Unstable

Write a Java program to solve this.

Solution:

```
import java.util.Scanner;

class HelloWorld {

   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter a number: ");

      int number = scanner.nextInt();

      int t = number; // Preserve the original number for future use if needed

      // Array to store the frequency of each digit (0-9)

      int[] ar = new int[10];

      // Initialize the frequency array

      for (int i = 0; i < 10; i++) {

         ar[i] = 0;

      }
```

```
        // Process each digit in the number using a while loop
        while (number > 0) {
            int digit = number % 10;
            ar[digit]++; // Increment the frequency of this digit
            number /= 10;
        }

        // Find the frequency of the first digit
        int firstDigit = t % 10;
        int frequency = ar[firstDigit];
        int flag = 0;

        // Check if all non-zero frequencies are equal to the frequency of the first digit
        for (int i = 0; i < 10; i++) {
            if (ar[i] != 0 && ar[i] != frequency) {
                flag = 1;
                System.out.println("Unstable");
                break;
            }
        }

        if (flag == 0) {
            System.out.println("Stable");
        }
    }
}
```

Question 5:

The encoded array is formed with the logic

arr[i]=arr[i]+arr[i+1]

i.e arr[0]=arr[0]+arr[1]

And the last element of both encoded array and original array is same.

Ex:1

Input:

Encoded array={7,6,8,16,12,3}

Output:

Original Array={2,5,1,7,9,3}

Ex:2

Input:

Encoded array={1,-1,3,12,5}

Output:

Original Array={-2,3,-4,7,5}

Write the java program to find the original array for the given encoded array?

Solutiom:

```
import java.util.*;
class HelloWorld {
   public static void main(String[] args) {

      int[] encoded = {7,6,8,16,12,3};
      int n = encoded.length;
      int[] original = new int[n];
      original[n-1] = encoded[n - 1];
```

```
        for (int i = n - 2; i >= 0; i--) {
            original[i] = encoded[i] - original[i + 1];
        }
        System.out.print("Original Array: ");
        for (int i = 0; i < original.length; i++) {
            System.out.print(original[i] + " ");
        }
        System.out.println();
    }
}
```

Question 6:

In a bank's security system, the sum of the even digits in a user's account number is used to generate a unique security token. Write a function to calculate this sum in Java.

Solution:

```
import java.util.*;
class HelloWorld {


   public static void main(String[] args) {
      int n;
      System.out.println("Enter the number:");
      Scanner sc=new Scanner(System.in);
      n=sc.nextInt();
      int sum=0;
      while(n>0)
      {
         int d=n%10;
         if(d%2==0)
         {
            sum=sum+d;
         }
         n=n/10;
      }
      System.out.println(sum);
   }
}
```

String Based Problems:

Question1:

Sarah was asked to form the output string from the given input string using the following two examples format. Solve this problem using java

Ex 1:

Input : Welcome

Output: emocWel

Ex 2:

Input : college

Output: egelcol

Solution:

```
import java.util.*;
class HelloWorld {
   public static void main(String[] args) {
      String s;
      System.out.println("Enter the String:");
      Scanner sc=new Scanner(System.in);
      s=sc.next();
      int l=s.length();
      for(int i=l-1;i>=l/2;i--){
        System.out.print(s.charAt(i));
                                    }
      for(int i=0;i<l/2;i++){
        System.out.print(s.charAt(i));
                                 }
}
```

}

Question2:

Ex 1:

In the class room teacher asked to find the resultant string from the input string based on the given two digits number. Solve this problem using java.

Ex:1

Input : Today is a nice day

41

Output: ecni yadTo

Ex 2:

Input : Welcome to the SRM college

51

Output: egelcol emocWel

Solution:

```
import java.util.*;
class HelloWorld {
    public static void string_process(String s)
  {
     int l=s.length();

     for(int i=l-1;i>=l/2;i--)
     {
       System.out.print(s.charAt(i));
     }

     for(int i=0;i<l/2;i++)
```

```
    {
      System.out.print(s.charAt(i));
    }
  }
   public static void main(String[] args) {
     String s;
     System.out.println("Enter the String:");
     Scanner sc=new Scanner(System.in);
     s=sc.nextLine();
     System.out.println("Enter the two digits to process:");
     int n=sc.nextInt();
     String words[]=s.split(" ");
     int d1=n/10;
     int d2=n%10;
     String w1=words[d1-1];
     String w2=words[d2-1];
    string_process(w1);
    System.out.print(" ");
    string_process(w2);
       }
 }
```

Input:

Enter the String:

SRM is the University

Enter the two digits to process:

41

Output:

ytisrUnive MRS

Question 3:

John received a secret message "SRM". By converting each letter to its corresponding number (A=1, B=2, ..., Z=26) and adding them up, what is the code?

Input: SRM

Output: 50 [S → 19 + R → 18 + M → 13]

Solution:

```
import java.util.*;
class HelloWorld {
   public static void main(String[] args) {

      String s;
      Scanner sc=new Scanner(System.in);
      s=sc.next();
      int sum=0;
      for(int i=0;i<s.length();i++)
      {
         sum=sum+(s.charAt(i)-'A'+1);
      }
      System.out.println(sum);
         }
}
```

Question 4:

In a puzzle, the term "SRM is the University" appears. If you assign values A=1, B=2, ..., Z=26, what is the sum of these values for the given words in the sentence (skip the spaces between the words)?

Solution:

```
import java.util.*;
class HelloWorld {
   public static void main(String[] args) {
       String s;
       System.out.println("Enter the Line:");
       Scanner sc=new Scanner(System.in);
       s=sc.nextLine();
      String upper_case=s.toUpperCase();

       String words[]=upper_case.split(" ");
       int sum=0;

        for(int j=0;j<words.length;j++)
       {
       for(int i=0;i<words[j].length();i++)
       {
          sum=sum+(s.charAt(i)-'A'+1);
       }
       }
       System.out.println(sum);
}
}
```

Question 5:

In a student grading system, a secret code is generated based on the student's name. Part of this code involves summing the ASCII values of all vowels in the student's name. Write a java code that takes a student's name as input and returns the sum of the ASCII values of the vowels

Solution:

```
import java.util.*;

class HelloWorld {


   public static void main(String[] args) {

      String s;

      System.out.println("Enter the String:");

      Scanner sc=new Scanner(System.in);

      s=sc.next();

      int sum=0;

      for(int i=0;i<s.length();i++)

      {

        if(s.charAt(i)=='A'|| s.charAt(i)=='E'|| s.charAt(i)=='I'||
s.charAt(i)=='O'||s.charAt(i)=='U'||s.charAt(i)=='a'||s.charAt(i)=='e'||s.charAt(i)=='i'||s.charAt(i)=='o'||
s.charAt(i)=='u')

        {

          sum=sum+(s.charAt(i));

        }

      }

      System.out.println(sum);

   }

}
```

Pattern based Problems:

Form this pattern using Java

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

Solution:

```
public class NumberPattern {
   public static void main(String[] args) {
      int rows = 5; // Number of rows in the pattern

      for (int i = 1; i <= rows; i++) { /
         for (int j = 1; j <= i; j++) {
            System.out.print(j + " ");
         }
         System.out.println();       }
   }
}
```

Question 2:

Form this pattern using Java

```
      1
    1 2 1
  1 2 3 2 1
1 2 3 4 3 2 1
```

Solution:

```
public class PyramidPattern {
    public static void main(String[] args) {
        int rows = 4; // Number of rows in the pattern
public class XPattern {
    public static void main(String[] args) {
        int rows = 4; // Number of rows in the pattern

        for (int i = 1; i <= rows; i++) {
            // Print leading spaces
            for (int j = i; j < rows; j++) {
                System.out.print("  ");
            }

            // Print 'x' with spaces in between
            for (int j = 1; j <= (2 * i - 1); j++) {
                System.out.print("x ");
            }

            // Move to the next line after each row
            System.out.println();
```

```
        }
    }
}
        for (int i = 1; i <= rows; i++) {
            // Print leading spaces
            for (int j = i; j < rows; j++) {
                System.out.print("  ");
            }

            // Print increasing part of the row
            for (int j = 1; j <= i; j++) {
                System.out.print(j + " ");
            }

            // Print decreasing part of the row
            for (int j = i - 1; j >= 1; j--) {
                System.out.print(j + " ");
            }

            // Move to the next line after each row
            System.out.println();
        }
    }
}
```

Question 3:

Form this pattern using Java

```
      x
    x x x
 x x  x  x x
x x x x  x  x x
```

Solution:

```
public class XPattern {
    public static void main(String[] args) {
        int rows = 4; // Number of rows in the pattern

        for (int i = 1; i <= rows; i++) {
            // Print leading spaces
            for (int j = i; j < rows; j++) {
                System.out.print("   ");
            }

            // Print 'x' with spaces in between
            for (int j = 1; j <= (2 * i - 1); j++) {
                System.out.print("x ");
            }

            // Move to the next line after each row
            System.out.println();
        }
```

```
    }
}
```