ASSIGNMENT-5 DATE-12/7/24

1. Write a program to find the square, cube of the given decimal number
   Sample Input:
      Given Number: 0.6
   Sample Output:
      Square Number: 0.36
      Cube Number:0.216

CODE:

```
public class DecimalNumber {
   public static void main(String[] args) {
      double givenNumber = 0.6;
      double squareNumber = givenNumber * givenNumber;
      double cubeNumber = givenNumber * givenNumber * givenNumber;
      System.out.println("Square Number: " + squareNumber);
      System.out.println("Cube Number: " + cubeNumber);
   }
}
```

OUTPUT:

2. Find the n^{th} odd number after n odd number
Sample Input: N : 7
Sample Output:
Hence the values printed for i are 1 , 3 , 5.

CODE:

```
public class OddNumbers {
   public static void main(String[] args) {
      int n = 7; // Sample Input
      int count = 0;
      int num = 1;

      while (count < n) {
         if (num % 2 != 0) {
            System.out.print(num + " ");
```

```
            count++;
        }

        num++;
    }
}
```

OUTPUT:

3. Program to find the frequency of each element in the array.

Sample Input & Output:

{1, 2, 8, 3, 2, 2, 2, 5, 1}

Pseudo:

```
Element | Frequency
-------------------------
1       |       2
2       |       4
8       |       1
3       |       1

4       |       1
```

CODE:

```
import java.util.HashMap;
public class Main {
    public static void main(String[] args) {
        int[] array = {1, 2, 8, 3, 2, 2, 2, 5, 1};
        HashMap<Integer, Integer> frequencyMap = new HashMap<>();
        for (int num : array) {
            frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);
        }
        System.out.println("Element | Frequency");
        System.out.println("-------------------------");
        frequencyMap.forEach((key, value) -> System.out.println(key + "       |       " +
value));
    }
}
```

OUTPUT:

4.Program to find whether the given number is Armstrong number or not

Sample Input:

Enter number: 153

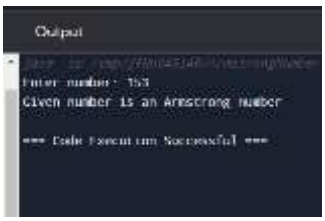
Sample Output:

Given number is Armstrong number

CODE:

```
public class ArmstrongNumber {
   public static void main(String[] args) {
      int number = 153;
      int originalNumber, remainder, result = 0;
      originalNumber = number;
      while (originalNumber != 0) {
         remainder = originalNumber % 10;
         result += Math.pow(remainder, 3);
         originalNumber /= 10;
      }
      if (result == number)
         System.out.println(number + " is an Armstrong number.");
      else
         System.out.println(number + " is not an Armstrong number.");
   }
}
```

OUTPUT:



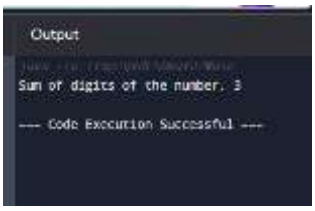
5. Write a program to find the sum of digits of N digit number (sum should be single digit)

CODE:

```
public class Main {
   public static void main(String[] args) {
      int number = 54897;
      int sum = 0;
      while (number > 0 || sum > 9) {
```

```
            if (number == 0) {
                number = sum;
                sum = 0;
            }
            sum += number % 10;
            number /= 10;
        }
        System.out.println("Sum of digits: " + sum);
    }
}
```

OUTPUT:



6.WRITE a program to find the square root of a perfect square number(print both the positive and negative values)

Sample Input:

Enter the number: 6561

Sample Output:

Square Root: 81, -81

CODE:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number: ");
        int number = scanner.nextInt();
        double squareRoot = Math.sqrt(number);
        System.out.println("Square Root: " + (int)squareRoot + ", " + -(int)squareRoot);
    }
}
```

OUTPUT:

7. Write a program to given an integer n, return true if it is a power of three. Otherwise, return false.

Input =27
Output= true
Explanation: $27=3^3$

CODE:

```
public class Main {
   public static void main(String[] args) {
      int n = 27;
      System.out.println(isPowerOfThree(n));
   }

   public static boolean isPowerOfThree(int n) {
      return n > 0 && 1162261467 % n == 0;
   }
}
```

OUTPUT:

8. Write a program to given a string paragraph and a string array of the banned words banned, return the most frequent word that is not banned. It is guaranteed there is at least one word that is not banned, and that the answer is unique.

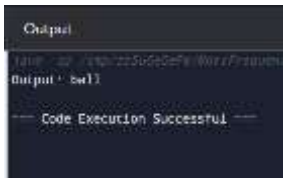
Input Paragraph=”Ram hit a ball, the hit ball flew far after it was hit”,
Banned = [hit]
Output=”Ball”

CODE:

```
import java.util.*;

public class MostFrequentWord {
   public String mostCommonWord(String paragraph, String[] banned) {
      Set<String> bannedWords = new HashSet<>(Arrays.asList(banned));
      Map<String, Integer> wordCount = new HashMap<>();
      String[] words = paragraph.replaceAll("[^a-zA-Z ]", "").toLowerCase().split("\\s+");

      for (String word : words) {
         if (!bannedWords.contains(word)) {
```

```
            wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);
        }
      }

      return Collections.max(wordCount.entrySet(),
Map.Entry.comparingByValue()).getKey();
   }

   public static void main(String[] args) {
      String paragraph = "Ram hit a ball, the hit ball flew far after it was hit";
      String[] banned = {"hit"};

      MostFrequentWord solution = new MostFrequentWord();
      String result = solution.mostCommonWord(paragraph, banned);

      System.out.println("Output: " + result);
   }
}
```

OUTPUT:



9. Write a program to given a fixed-length integer array arr, duplicate each occurrence of zero, shifting the remaining elements to the right.
Input: arr = [1, 0, 2, 3, 0, 4, 5, 0]
Output: [1, 0, 0, 2, 3, 0, 0, 4]

CODE:

```
public class DuplicateZeros {
   public static void duplicateZeros(int[] arr) {
      int zeros = 0;
      for (int num : arr) {
         if (num == 0) {
            zeros++;
         }
      }

      int n = arr.length;
      for (int i = n - 1, j = n + zeros - 1; i < j; i--, j--) {
         if (arr[i] == 0) {
```

```
            if (j < n) {
                arr[j] = 0;
            }
            j--;
        }
        if (j < n) {
            arr[j] = arr[i];
        }
    }
}

public static void main(String[] args) {
    int[] arr = {1, 0, 2, 3, 0, 4, 5, 0};
    duplicateZeros(arr);
    System.out.print("Output: [");
    for (int i = 0; i < arr.length; i++) {
        System.out.print(arr[i]);
        if (i < arr.length - 1) {
            System.out.print(", ");
        }
    }
    System.out.println("]");
}
}
```

OUTPUT:

10.Write a program to given an array nums containing n distinct numbers in the range [0, n], return the only number in the range that is missing from the array.

Input nums = [3, 0, 1]

Output: 2

.

CODE:

```
public class MissingNumber {
    public int missingNumber(int[] nums) {
        int n = nums.length;
        int total = n * (n + 1) / 2;
        for (int num : nums) {
            total -= num;
        }
```

```
        return total;
    }

    public static void main(String[] args) {
        int[] nums = {3, 0, 1};
        MissingNumber mn = new MissingNumber();
        System.out.println("Missing number: " + mn.missingNumber(nums));
    }
}
```

OUTPUT: