

12/7/24

ASSIGNMENT -5

1. Write a program to find the square, cube of the given decimal number

Sample Input:

Given Number: 0.6

Sample Output:

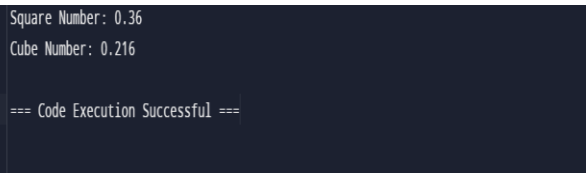
Square Number: 0.36

Cube Number:0.216

Test cases:

1. 12
2. 0
3. -0.5
4. 14.25
5. -296

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



```
Square Number: 0.36
Cube Number: 0.216

=== Code Execution Successful ===
```

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.

Test cases:

1. N = 0
2. N = -6
3. N = 2021
4. N = -14.5
5. N = -196

```
public class OddNumbers {
    public static void main(String[] args) {
        int N = 7;
        int count = 0;
        int i = 1;
```

```

while (count < N) {
    if (i % 2 != 0) {
        System.out.print(i + " ");
        count++;
    }
    i++;
}
}
}

```

```

1 3 5 7 9 11 13
--- Code Execution Successful ---

```

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

```

```

public class Frequency {

```

```

    public static void main(String[] args) {

```

```

        //Initialize array

```

```

        int [] arr = new int [] {1, 2, 8, 3, 2, 2, 2, 5, 1};

```

```

        //Array fr will store frequencies of element

```

```

        int [] fr = new int [arr.length];

```

```

        int visited = -1;

```

```

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

```

```

            int count = 1;

```

```

            for(int j = i+1; j < arr.length; j++){

```

```

                if(arr[i] == arr[j]){

```

```

                    count++;

```

```

                    //To avoid counting same element again

```

```

                    fr[j] = visited;

```

```

                }

```

```

            }

```

```

            if(fr[i] != visited)

```

```

                fr[i] = count;

```

```

        }

```

```

//Displays the frequency of each element present in array
System.out.println("-----");
System.out.println(" Element | Frequency");
System.out.println("-----");
for(int i = 0; i < fr.length; i++){
    if(fr[i] != visited)
        System.out.println("    " + arr[i] + "    |    " + fr[i]);
}
System.out.println("-----");
}
}

```

```

java -cp /tmp/ckklog9jfb/Frequency
Element | Frequency
-----
1 | 2
2 | 4
8 | 1
3 | 1
5 | 1
-----
=== Code Execution Successful ===

```

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

Test cases:

1. 370
2. 1
3. 371
4. 145678
5. 0.21345

```

import java.util.Scanner;
public class ArmstrongNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number: ");
        int number = sc.nextInt();
        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("Given number is Armstrong number");
    }
}

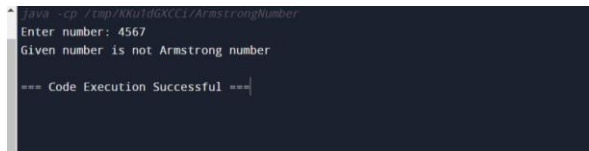
```

```

        else
            System.out.println("Given number is not Armstrong number");

        sc.close();
    }
}

```



```

java -cp /tmp/xx0106cc17/ArmstrongNumber
Enter number: 4567
Given number is not Armstrong number

*** Code Execution Successful ***

```

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

Sample Input:

Enter N value: 3

Enter 3 digit numbers: 143

Test cases:

1. N = 2, 158
2. N = 3, 14
3. N = 4, 0148
4. N = 1, 0004
5. N = 4, 7263

```

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter N value: ");
        int n = input.nextInt();
        System.out.print("Enter " + n + " digit number: ");
        int num = input.nextInt();
        int sum = 0;

        while (num > 0 || sum > 9) {
            if (num == 0) {
                num = sum;
                sum = 0;
            }
            sum += num % 10;
            num /= 10;
        }

        System.out.println("Sum of digits: " + sum);
    }
}

```

```
java -cp ./tmp/bn.0ybbP1N/Main
Enter N value: 3
Enter 3 digit number: 234
Sum of digits: 9

=== Code Execution Successful ===
```

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

Test cases:

1. 1225

2. 9801

3. 1827

4. -100

5. 0

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

int number = input.nextInt();

double squareRoot = Math.sqrt(number);

System.out.println("Square Root: " + (int)squareRoot + ", " + -(int)squareRoot);

}

}

```
Enter the number: 64
Square Root: 8, -8

=== Code Execution Successful ===
```

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: $27 = 3^3$

Test cases:

1. 12

2. abc@45

3. 1827

4. -100

5. 0

public class PowerOfThree {

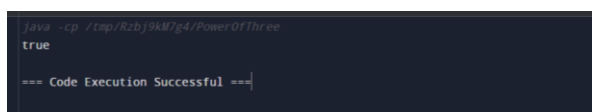
public boolean isPowerOfThree(int n) {

if (n <= 0) {

```

        return false;
    }
    while (n % 3 == 0) {
        n /= 3;
    }
    return n == 1;
}
public static void main(String[] args) {
    PowerOfThree powerOfThree = new PowerOfThree();
    System.out.println(powerOfThree.isPowerOfThree(27));
}

```



```

java -cp ./tmp/Rzbj9kM7p4/PowerOfThree
true
=== Code Execution Successful ===

```

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"

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();
        System.out.println(solution.mostCommonWord(paragraph, banned));
    }
}

```



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]

Explanation: After calling your function, the input array is modified to [1, 0, 0, 2, 3, 0, 0,]

```
public class DuplicateZeros {
    public static void duplicateZeros(int[] arr) {
        int n = arr.length;
        int count = 0
        for (int i = 0; i < n; i++) {
            if (i + count >= n - 1) {
                break;
            }
            if (arr[i] == 0) {
                if (i + count == n - 2) {
                    arr[n - 1] = 0;
                    n--;
                    break;
                }
                count++;
            }
        }

        // Second pass: fill from the end
        for (int i = n - 1 - count; i >= 0; i--) {
            if (arr[i] == 0) {
                arr[i + count] = 0;
                count--;
                arr[i + count] = 0;
            } else {
                arr[i + count] = arr[i];
            }
        }
    }

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

```
1 0 0 2 3 0 0 4  
=== Code Execution Successful ===
```

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

Explanation: n = 3 since there are 3 numbers, so all numbers are in the range [0, 3]. 2 is the missing number in the range since it does not appear in nums

```
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));  
    }  
}
```

```
Missing number: 2  
=== Code Execution Successful ===
```