Problem : Two Sum Problem

Problem Statement

Write a function `twoSum(nums, target)` that takes an array of integers (`nums`) and an integer (`target`) as input.
The goal is to find **two numbers in the array** that add up exactly to the `target`.

- Your function should return the **indices** of the two numbers.
- **You may not use the same element twice.**
- If no two numbers sum up to the target, the function should return an empty array (`[]`).
- You must also **write a `main` method** that performs the following steps:
    1. **Takes input** for the array and the target value from the user.
    2. **Validates the input array format**, ensuring it contains only numbers separated by commas (with optional spaces).
    3. **Calls the `twoSum()` function** with the validated array and target value.
    4. **Displays the result** appropriately.

Example 1: Solution Found

Input:

```
nums = [2, 7, 11, 15]
target = 9
```

Output:

```
[0,1]
```

Explanation: Because nums[0] (which is 2) + nums[1] (which is 7) equals 9, we return the indices [0, 1].

Example 2: No Solution Found

Input:

```
nums = [3, 5, 8]
target = 20
```

Output:

```
[]
```

Explanation: No two numbers in the array sum to 20, so we return an empty array.

Constraints

You must find the solution with a time complexity between $O(n^2)$ and $O(n)$.

Code

```
import java.util.Scanner;
import java.util.Arrays;
import java.util.HashMap;

class Solution2{
    // Solution in O(n)
    public static int[] twoSum(int[] nums, int target) {
        HashMap<Integer, Integer> mpp = new HashMap<>();
        int sub = 0;
        for(int i = 0; i < nums.length; ++i) {
            sub = target - nums[i];
            if(mpp.containsKey(sub))
                return new int[]{mpp.get(sub), i};
            mpp.put(nums[i], i);
        }
        return new int[]{};
    }

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

        String inputRegex = "^\\s*(-?\\d{1,9})(\\s*,\\s*-?\\d{1,9})*\\s*$";
        /* valid inputs
            "123, 456,789",
            "1,2,3",
            "  12 , 34 , 56  "
        */

        System.out.println();
        System.out.println("Enter the integer array \n For eg.
1,2,6,3,56,45....");

        String inputArr = sc.nextLine();

        System.out.println();
        if(inputArr.matches(inputRegex)){
            System.out.println("The input array is corect.");
        }
        else{
            System.out.println("The input array is in-corect. Existing the
program...");
            return;
        }

        String[] inputs = inputArr.split(",");

        int[] arr = Arrays.stream(inputs)
                    .map(String::trim)  //space removal
                    .mapToInt(Integer::parseInt) // cast to int
                    .toArray();

        System.out.println();
        System.out.println("Given int arr "+ Arrays.toString(arr));

        System.out.println();
        System.out.println("Enter the target (the specific sum you want to
find)");
        int target = sc.nextInt();

        int[] result = twoSum(arr,target);
        String message;
        if(result.length>0){
            message = String.format("[%d,%d]",result[0],result[1]);
        }
        else{
            message= "[]";
        }
        System.out.println(message);
    }
}
```

Terminal Output

```
$ java Solution2

Enter the integer array
 For eg. 1,2,6,3,56,45....
2, 7, 11, 15

The input array is corect.

Given int arr [2, 7, 11, 15]

Enter the target (the specific sum you want to find)
9
[0,1]
```

Problem: Maximum Subarray Sum (Kadane’s Algorithm)

Problem Statement

Given an integer array `arr[]`, find the **subarray** (containing at least one element) which has the **maximum possible sum**, and return that sum.

> Note: A subarray is a **continuous part** of an array.

🧠 Examples

Example 1:

```
Input: arr[] = [2, 3, -8, 7, -1, 2, 3]
Output: 11
Explanation: The subarray [7, -1, 2, 3] has the largest sum = 11.
```

Example 2:

```
Input: arr[] = [-2, -4]
Output: -2
Explanation: The subarray [-2] has the largest sum -2.
```

Example 3:

```
Input: arr[] = [5, 4, 1, 7, 8]
Output: 25
Explanation: The subarray [5, 4, 1, 7, 8] has the largest sum 25.
```

Code

```
import java.util.Scanner;
import java.util.Arrays;

class Solution2{

    public static int maximumSubArraySum(int[] arr){
        int res = arr[0];
        int maxEnding = arr[0];

        // Kadane's appraoch O(n)
        for(int i=1;i<arr.length;i++){
            maxEnding = Math.max(arr[i] ,maxEnding + arr[i]);

            res = Math.max(res,maxEnding);
        }

        return res;

    }

    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        String inputRegex = "^\\s*(-?\\d{1,9})(\\s*,\\s*-?\\d{1,9})*\\s*$";
        /* valid inputs
            "123, 456,789",
            "1,2,3",
            "   12 , 34 , 56   "
        */

        System.out.println();
        System.out.println("Enter the integer array \n For eg.
1,2,6,3,56,45....");

        String inputArr = sc.nextLine();

        System.out.println();
        if(inputArr.matches(inputRegex)){
            System.out.println("The input array is corect.");
        }
        else{
            System.out.println("The input array is in-corect. Exiting the
program...");
            return;
        }

        int[] arr = Arrays.stream(inputArr.split(","))
                    .map(String::trim)
                    .mapToInt(Integer::parseInt)
                    .toArray();


        int result = maximumSubArraySum(arr);

        System.out.println("The maximum sub-array sum is : "+result);
    }
}
```

Terminal Output

```
$ java Solution2

Enter the integer array
```

```
 For eg. 1,2,6,3,56,45....
5, 4, 1, 7, 8

The input array is corect.
The maximum sub-array sum is : 25
```

Problem: Merge Sort

Problem Statement

Implement the **Merge Sort** algorithm to sort an array of integers in ascending order.

Merge Sort is a **divide and conquer** algorithm that works as follows:

1. Divide the array into two halves.
2. Recursively sort each half.
3. Merge the two sorted halves into a single sorted array.

Your task is to write a function that takes an array of integers and returns a sorted array.

Example 1:

```
Input: arr[] = [5, 3, 8, 4, 2]
Output: [2, 3, 4, 5, 8]
```

Code

```java
import java.util.Scanner;
import java.util.Arrays;

class Solution{
    public static void merge(int[] arr, int low, int mid, int high){
        int n = mid - low + 1;
        int m = high - mid;

        int left[] = new int[n];
        int right[] = new int[m];

        //copy to temp arr
        for(int i = 0; i<n; i++){
            left[i] = arr[low + i];
        }

        for(int i = 0; i<m; i++){
            right[i] = arr[mid + 1 + i];
        }

        int i = 0, j=0, k = low;
        while(i< n && j< m){
            if(left[i] <= right[j]){
                arr[k] = left[i];
                i++;
            }
            else{
                arr[k] = right[j];
                j++;
            }
            k++;
        }

        // residual elements
        while(i<n){
            arr[k] = left[i];
            i++;
            k++;
        }

        while(j<m){
            arr[k] = right[j];
            j++;
            k++;
        }
    }

    public static void mergeSort(int[] arr, int low, int high){
        if(low<high){
            int mid = low + (high-low)/2;

            mergeSort(arr,low,mid);
            mergeSort(arr,mid+1,high);

            merge(arr, low, mid, high);
        }
    }

    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        String inputRegex = "^\\s*(-?\\d{1,9})(\\s*,\\s*-?\\d{1,9})*\\s*$";

        System.out.println("Enter the integer array \n For eg.
1,2,6,3,56,45....");
        String inputArr = sc.nextLine();

        if(inputArr.matches(inputRegex)){
            System.out.println("The input array is corect.");
        }
        else{
            System.out.println("The input array is in-corect. Exiting the
program...");
            return;
        }

        int[] arr = Arrays.stream(inputArr.split(","))
                        .map(String::trim)
                        .mapToInt(Integer::parseInt)
                        .toArray();

        System.out.println("Input array before merge sort"+Arrays.toString(arr));
        mergeSort(arr,0,arr.length-1);
        System.out.println("Input array after merge sort"+Arrays.toString(arr));
    }
}
```

Terminal Output

```
$ java Solution
Enter the integer array
 For eg. 1,2,6,3,56,45....
5, 3, 8, 4, 2
The input array is corect.
Input array before merge sort[5, 3, 8, 4, 2]
Input array after merge sort[2, 3, 4, 5, 8]
```

Problem : Largest Number

Problem Statement

Given a list of non-negative integers nums, arrange them such that they form the largest number and return it.

Since the result may be very large, so you need to return a string instead of an integer.

Write a function, `largestNumber(int[] arr)`, that finds the **largest number** among an array of integers.

- You must also write a **main** function that:
  - Takes user input for an array of non-negative integers (comma- separated).
    - Example input: 2,59,9,4,10,89
  - Validates the input using a regular expression.
  - Calls the largestNumber() method and prints the result.

Example 1:

```
Input: nums = [10,2]
Output: "210"
```

Example 2:

```
Input: nums = [3,30,34,5,9]
Output: "9534330"
```

Code

```
import java.util.*;

class Solution{

    public static String largestNumber(int[] nums){
        String[] strNums = new String[nums.length];
        for (int i = 0; i < nums.length; i++) {
            strNums[i] = String.valueOf(nums[i]);
        }

        Arrays.sort(strNums, (a, b) -> (b + a).compareTo(a + b));

        if (strNums[0].equals("0")) return "0";

        StringBuilder sb = new StringBuilder();
        for (String s : strNums)
```

```
            sb.append(s);

        return sb.toString();
    }

    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        String inputRegex = "^\\s*(\\d{1,9})(\\s*,\\s*\\d{1,9})*\\s*$";

        System.out.println("Enter the positive integer array \n For eg.
1,2,6,3,56,45....");
        String inputArr = sc.nextLine();

        if(inputArr.matches(inputRegex)){
            System.out.println("The input array is corect.");
        }
        else{
            System.out.println("The input array is in-corect. Exiting the
program...");
            return;
        }

        int[] arr = Arrays.stream(inputArr.split(","))
                    .map(String::trim)
                    .mapToInt(Integer::parseInt)
                    .toArray();

        System.out.println("Input array: "+Arrays.toString(arr));
        String result = largestNumber(arr);
        System.out.println("Result : "+result);
    }
}
```

Terminal Output

```
$ java Solution
Enter the positive integer array
 For eg. 1,2,6,3,56,45....
3,30,34,5,9
The input array is corect.
Input array: [3, 30, 34, 5, 9]
Result : 9534330
```

Problem: Product of Array Except Self

Problem Statement

Given an integer array nums, return an array answer such that answer[i] is equal to the product of all the elements of nums except nums[i].

Do not use division, and solve in O(n) time.

If any array element is `0`, then the division may cause `divide by 0` exceoption.

Write a function `productExceptSelf(nums)` that takes an array of integers (`nums`) as input.
The goal is to return an array `answer` such that `answer[i]` is equal to the **product of all elements in `nums` except `nums[i]`**.

- **Do not use division**.
- Solve the problem in **O(n) time**.
- If the array contains `0`, handle it properly to avoid division errors.

You must also **write a `main` method** that performs the following steps:

1. **Takes input** for the array from the user.
2. **Validates the input array format**, ensuring it contains only numbers separated by commas (with optional spaces).
3. **Calls the `productExceptSelf()` function** with the validated array.
4. **Displays the result** appropriately.

Example 1

```
Input: arr[] = [10, 3, 5, 6, 2]
Output: [180, 600, 360, 300, 900]
Explanation:

For i=0, res[i] = 3 * 5 * 6 * 2 is 180.
For i = 1, res[i] = 10 * 5 * 6 * 2 is 600.
For i = 2, res[i] = 10 * 3 * 6 * 2 is 360.
For i = 3, res[i] = 10 * 3 * 5 * 2 is 300.
For i = 4, res[i] = 10 * 3 * 5 * 6 is 900.
```

Example 2

```
Input: arr[] = [12, 0]
Output: [0, 12]
Explanation:

For i = 0, res[i] = 0.
For i = 1, res[i] = 12.
```

Code

```
import java.util.Scanner;
import java.util.Arrays;

class Solution{

    public static int[] productExceptSelf(int arr[]){
        int n = arr.length;

        int[] prefixProduct = new int[n];
        int[] suffixProduct = new int[n];
        int[] result = new int[n];

        prefixProduct[0] = 1;
        for(int i=1; i<n; i++){
            prefixProduct[i] = arr[i-1] * prefixProduct[i-1];
        }

        suffixProduct[n-1]=1;
        for(int j=n-2; j>=0; j--){
            suffixProduct[j] = arr[j+1] * suffixProduct[j+1];
        }

        for(int i=0; i<n; i++){
            result[i] = prefixProduct[i] * suffixProduct[i];
        }

        return result;
    }

    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        String inputRegex = "^\\s*(-?\\d{1,9})(\\s*,\\s*-?\\d{1,9})*\\s*$";

        System.out.println("Enter the integer array \n For eg.
1,2,6,3,56,45....");
        String inputArr = sc.nextLine();

        if(inputArr.matches(inputRegex)){
            System.out.println("The input array is corect.");
        }
        else{
            System.out.println("The input array is in-corect. Exiting the
program...");
            return;
        }
        System.out.println();
        int[] arr = Arrays.stream(inputArr.split(","))
                        .map(String::trim)
                        .mapToInt(Integer::parseInt)
                        .toArray();

        System.out.println("Input array "+Arrays.toString(arr));

        int result[] = productExceptSelf(arr);

        System.out.println("Result "+Arrays.toString(result));
    }
}
```

Terminal Output

```
$ java Solution
Enter the integer array
 For eg. 1,2,6,3,56,45....
10, 3, 5, 6, 2
The input array is corect.

Input array [10, 3, 5, 6, 2]
Result [180, 600, 360, 300, 900]
```

Problem: Find All Duplicates in an Array

Problem Statement

Given an array of integers where 1 ≤ a[i] ≤ n (n = size of array), some elements appear twice and others once.

Find all the elements that appear twice.

Example 1:

```
Input: [4,3,2,7,8,2,3,1]
Output: [2,3]
```

Example 2:

```
Input: [1,1,2]
Output: [1]
```

Code

```
import java.util.*;

class Solution{

    public static ArrayList<Integer> findDuplicateElements(int arr[]){
        ArrayList<Integer> result = new ArrayList<>();
        HashMap<Integer,Integer> map = new HashMap<>();

        for(int num : arr){
            map.put(num , map.getOrDefault(num,0)+1);
        }

        for(var pair : map.entrySet()){
            if(pair.getValue()!=1){
                result.add(pair.getKey());
            }
        }

        return result;
    }

    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        String inputRegex = "^\\s*(-?\\d{1,9})(\\s*,\\s*-?\\d{1,9})*\\s*$";

        System.out.println("Enter the integer array \n For eg.
1,2,6,3,56,45....");
        String inputArr = sc.nextLine();

        if(inputArr.matches(inputRegex)){
            System.out.println("The input array is corect.");
        }
        else{
            System.out.println("The input array is in-corect. Exiting the
program...");
            return;
        }
        System.out.println();
        int[] arr = Arrays.stream(inputArr.split(","))
                    .map(String::trim)
                    .mapToInt(Integer::parseInt)
                    .toArray();

        System.out.println("Input array "+Arrays.toString(arr));

        ArrayList<Integer> result = findDuplicateElements(arr);

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

Terminal Output

```
$ java Solution
Enter the integer array
 For eg. 1,2,6,3,56,45....
4,3,2,7,8,2,3,1
The input array is corect.

Input array [4, 3, 2, 7, 8, 2, 3, 1]
Result [2, 3]
```

Problem 1: Longest Common Prefix

Problem Statement

Write a function, `longestCommonPrefix(String[] strs)`, that finds the **longest common prefix** string among an array of strings.

If there is **no common prefix**, return an empty string `""`.

- You must also write a **main** function that:
  - Takes user input for an array of strings (comma- separated).
    - Example input: apple, application, appstore
  - Validates the input using a regular expression.
  - Splits and trims the strings into an array.
  - Calls the longestCommonPrefix() method and prints the result.

Example 1: Common Prefix Found

```
Input: strs = ["flower", "flow", "flight"]

Output: "fl"

Explanation: The longest common prefix among the three strings is "fl".
```

Example 2: No Common Prefix Found

```
Input: strs = ["dog", "racecar", "car"]

Output: ""

Explanation: There is no common prefix among the input strings.
```

Code

```
import java.util.Scanner;
import java.util.Arrays;
class Solution {
    public static String longestCommonPrefix(String[] strs) {
        String longest = strs[0];

        for(int i=1;i<strs.length;i++){
            if(longest.length()==0){
                return longest;
            }
            for(int j=0; j<Math.min(longest.length(),strs[i].length()); j++){
                if(longest.charAt(j)!=strs[i].charAt(j)){
                    longest  = longest.substring(0,j);
                    break;
                }
            }
            longest =
longest.substring(0,Math.min(longest.length(),strs[i].length()));
        }

        return longest;
    }

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

        String inputRegex = "^\\s*\\w+(\\s*,\\s*\\w+)*\\s*$";
        /* valid inputs
           "apple,banana,cherry" ✅
            "apple, banana, cherry " ✅
            " apple , banana " ✅
        */

        System.out.println();
        System.out.println("Enter the string array (comma-separated, optional
spaces). \nFor example: apple, banana, cherry");

        String inputArr = sc.nextLine();

        System.out.println();
        if(inputArr.matches(inputRegex)){
            System.out.println("The input array is corect.");
        }
        else{
            System.out.println("The input array is in-corect. Exiting the
program...");
            return;
        }

        String[] arr = Arrays.stream(inputArr.split(","))
                        .map(String::trim)
                        .toArray(String[]::new);

        String result = longestCommonPrefix(arr);

        System.out.println();
        System.out.println();
        System.out.println("The most common longest prefix is : '"+result+"'");
    }
}
```

Terminal Output

```
$ java Solution

Enter the string array (comma-separated, optional spaces).
For example: apple, banana, cherry
flower,flow,flight

The input array is corect.

The most common longest prefix is : 'fl'
```

Problem : Anagram

Problem Statement

Given two non-empty strings s1 and s2, consisting only of lowercase English letters, determine whether they are anagrams of each other or not.

Two strings are considered anagrams if they contain the same characters with exactly the same frequencies, regardless of their order.

Write a function `checkAnagram(String s1, String s2)` that returns:

- `true` if both strings are anagrams.
- `false` otherwise.
- You must also write a **main** function that:
  - Takes user input for two strings.
  - Sanitizes the input and ensures the string contains only lowercase English letters.
  - Calls the `checkAnagram()` function and prints the result.

Example 1:

> **Input:** s1 = "geeks" s2 = "kseeg"
>
> **Output:** true
>
> **Explanation:** Both the string have same characters with same frequency. So, they are anagrams.

Example 2:

> **Input:** s1 = "allergy", s2 = "allergyy"
>
> **Output:** false
>
> **Explanation:** Although the characters are mostly the same, s2 contains an extra 'y' character. Since the frequency of characters differs, the strings are not anagrams.

Example 3:

> **Input:** s1 = "listen", s2 = "lists"
>
> **Output:** false
>
> **Explanation:** The characters in the two strings are not the same — some are missing or extra. So, they are not anagrams.

Code

```java
import java.util.Scanner;
import java.util.HashMap;
```

```
class Solution{
    public static boolean checkAnagram(String str1, String str2){

        if(str1.length() != str2.length()) return false;

        HashMap<Character,Integer> freqMap = new HashMap<>();

        for(char ch : str1.toCharArray()){
            freqMap.put(ch, freqMap.getOrDefault(ch,0)+1);
        }

        for(char ch : str2.toCharArray()){
            freqMap.put(ch, freqMap.getOrDefault(ch,0)-1);
        }

        for(var data : freqMap.entrySet()){
            if(data.getValue()!=0){
                return false;
            }
        }

        return true;
    }

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

        String str1,str2;

        System.out.println("Enter the string-1");
        str1 = sc.nextLine().trim().toLowerCase();
        System.out.println("Enter the string-2");
        str2 = sc.nextLine().trim().toLowerCase();
        // System.out.println("Input String "+str1+" "+str2);

        boolean result = checkAnagram(str1,str2);
        System.out.println(result);
        System.out.printf("Given two strings '%s', '%s' %s%s", str1, str2, result
? "are" : "are not", " Anagram");

        return;
    }
}
```

Terminal Output

```
$ java Solution
Enter the string-1
tea
Enter the string-2
```

```
ate
true
```

Problem : Roman to Integer

Problem Statement

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

```
Symbol Value

I = 1

V = 5

X = 10

L = 50

C = 100

D = 500

M = 1000
```

For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

I can be placed before V (5) and X (10) to make 4 and 9. X can be placed before L (50) and C (100) to make 40 and 90. C can be placed before D (500) and M (1000) to make 400 and 900. Given a roman numeral, convert it to an integer.

Write a function `romanToInt(String s)` that converts a **Roman numeral** into an **integer**.

If the given string is **not a valid Roman numeral**, your program should print an appropriate message and exit.

You must also write a **main** function that:

- Takes user input for a Roman numeral string.
- Validates whether it is a **valid Roman numeral**.
- Calls the `romanToInt()` function and prints the integer value.

Example 1:

> **Input:** s = "III"
>
> **Output:** 3
>
> **Explanation:** III = 3.

Example 2:

> **Input:** s = "LVIII"
>
> **Output:** 58
>
> **Explanation:** L = 50, V= 5, III = 3.

Example 3:

> **Input:** s = "MCMXCIV"
>
> **Output:** 1994
>
> **Explanation:** M = 1000, CM = 900, XC = 90 and IV = 4.

Code

```
import java.util.Scanner;

class Solution {

    public static boolean isValidRoman(String s) {
    if (s == null || s.isEmpty()) return false;

    // Roman numerals can only use these characters
    if (!s.matches("^[IVXLCDM]+$")) return false;

    // Strict Roman numeral validation using regex
    // Covers subtractive rules like IV, IX, XL, XC, CD, CM
    String romanRegex =
        "^M{0,3}" +                // Thousands - 0 to 3000
        "(CM|CD|D?C{0,3})" +       // Hundreds - 900 (CM), 400 (CD), 0-300 (C,
CC, CCC)
        "(XC|XL|L?X{0,3})" +       // Tens - 90 (XC), 40 (XL), 0-30 (X, XX, XXX)
        "(IX|IV|V?I{0,3})$";       // Ones - 9 (IX), 4 (IV), 0-3 (I, II, III)

    return s.matches(romanRegex);
}

    public static int value(char c){
        switch(c){
            case 'I': return 1;
            case 'V': return 5;
            case 'X': return 10;
            case 'L': return 50;
            case 'C': return 100;
            case 'D': return 500;
            case 'M': return 1000;
            default: return 0;
        }
    }
    public static int romanToInt(String s) {
        s = s.toUpperCase();
        int sum = 0;
        int length = s.length();

        for(int i=0;i<length;i++){
            int currValue = value(s.charAt(i));

            if(i+1 < length && currValue < value(s.charAt(i+1))){
                sum-=currValue;
            }
            else{
                sum+=currValue;
            }
        }
        return sum;
    }

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

        System.out.println("Enter you roman string / roman number.");
        String s = sc.nextLine().trim().toUpperCase();

        if(!isValidRoman(s)){
            System.out.println("Invalid Roman Literal. Quiting the program ...");
            return;
        }

        int result = romanToInt(s);

        System.out.println("The computed value of your roman number is : 
"+result);
    }
}
```

Terminal Output

```
$ java Solution
Enter you roman string / roman number.
MCMXCIV
The computed value of your roman number is :  1994
```

Problem : Longest Substring Without Repeating Characters

Problem Statement

Given a string s, find the length of the longest substring that contains no repeating characters. A substring is a contiguous sequence of characters within the string.

Write a function `Longest_Substring_Without_Repeating_Characters(String s)` that finds `Longest Substring Without Repeating Characters` from the given string.

You must also write a **main** function that:

- Takes user input for a string.
- Calls the `Longest_Substring_Without_Repeating_Characters()` function and prints the result value.

Example 1:

> **Input:** s = "abcabcbb"
>
> **Output:** substring = abc , size = 3
>
> **Explanation:** The answer is "abc", with a length of 3.

Example 2:

> **Input:** s = "pwwkew"
>
> **Output:** substring = wke , size = 3
>
> **Explanation:** The longest substring without repeating characters is "wke".

Code

```java
import java.util.Scanner;
import java.util.HashMap;

class Solution{

    public static String Longest_Substring_Without_Repeating_Characters(String str){
        int n = str.length();
        HashMap<Character,Integer> charMap = new HashMap<>();
        int maxLen = 0;
        int left = 0;
        String subStr="";

        for(int i=0; i<n; i++){
            char c= str.charAt(i);
            if(charMap.containsKey(c) && charMap.get(c)>=left){
                left = charMap.get(c)+1;
            }

            charMap.put(c,i);

            int currLen = i - left + 1;

            if(currLen>maxLen){
                maxLen = currLen;
                subStr = str.substring(left,i+1);
            }
        }

        return subStr;
    }

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

        System.out.println("Enter a String.");
        String s = sc.nextLine().trim();

        String result = Longest_Substring_Without_Repeating_Characters(s);
        System.out.printf("substring = '%s', size = %d",result, result.length());
    }
}
```

Terminal Output

```
$ java Solution
Enter a String.
pwwkew
substring = 'wke', size = 3
```

Problem : Group Anagrams

Problem Statement

Given an array of strings strs, group all the anagrams together.

Two strings are anagrams if they contain the same characters in different order.

Write a function `groupAnagram(String[] strArr)` that finds `Grouped Anagram` from the given string array.

You must also write a **main** function that:

- Takes user input for a string.
- Calls the `groupAnagram()` function and prints the result value.

Example 1:

> **Input:** strs = ["eat","tea","tan","ate","nat","bat"]
>
> **Output:** [["eat","tea","ate"],["tan","nat"],["bat"]]
>
> **Explanation:**

Example 1:

> **Input:** strs = [""]
>
> **Output:** [[""]]
>
> **Explanation:**

Code

```java
import java.util.*;

class Solution{
    public static String generateHashKey(String s){
        int arr_size = 26; // length of english characters;

        StringBuilder hashString = new StringBuilder();
        int[] frequency = new int[arr_size];

        for(char c : s.toCharArray()){
            frequency[c - 'a'] += 1;  //frequency of each char in str
        }

        for(int i=0; i<arr_size; i++){
            hashString.append(frequency[i]);
            hashString.append("$");
        }

        return hashString.toString();
    }

    public static ArrayList<ArrayList<String>> groupAnagram(String[] strArr){
        ArrayList<ArrayList<String>> result = new ArrayList<>();
        HashMap<String, Integer> map = new HashMap<>();

        for(int i = 0; i<strArr.length; i++){
            String key = generateHashKey(strArr[i]);

            if(!map.containsKey(key)){
                map.put(key,result.size());   // map keep ths index of the keys
                result.add(new ArrayList<>());  // at every index we keep the
arraylist of same key,
            }

            result.get(map.get(key)).add(strArr[i]);  // store the string in its
respective arraylist index
        }

        return result;
    }

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

        System.out.println("Enter string seperated by ',' comma.  Eg. abc, cdb,
xyz ....");
        String[] str = Arrays.stream(sc.nextLine().trim().split(","))
                        .map(String::trim)
                        .toArray(String[]::new);

        ArrayList<ArrayList<String>> result = groupAnagram(str);

        System.out.println(result);
    }
}
```

Terminal Output

```
$ java Solution
Enter string seperated by ',' comma.  Eg. abc, cdb, xyz ....
eat,tea,tan,ate,nat,bat
[[eat, tea, ate], [tan, nat], [bat]]
```

Problem: Search a 2D Matrix

Problem Statement

You are given an `m x n` integer matrix `matrix` with the following properties:

1. Each row is sorted in **non-decreasing order**.
2. The **first integer of each row** is greater than the **last integer of the previous row**.
3. You must also **write a `main` method** that performs the following steps:
    1. **Takes input** for the matrix and the target value from the user.
    2. **Validates the input matrix format**, ensuring it contains only numbers separated by commas (with optional spaces).
    3. **Calls the `searchMatrix()` function** with the validated array and target value.
    4. **Displays the result** appropriately.

Given an integer `target`, return `true` if `target` is present in the matrix, or `false` otherwise.

Example 1 : Target Exists

Input:

```
matrix = [
  [1, 3, 5, 7],
  [10, 11, 16, 20],
  [23, 30, 34, 60]
]
target = 3
```

Output:

```
Result: Target found!
```

Example 2 : Target Doesn't Exists

Input:

```
matrix = [
  [1, 3, 5, 7],
  [10, 11, 16, 20],
  [23, 30, 34, 60]
]
target = 13
```

Output:

```
Result: Target Not found!
```

Constraints:

- You must write a solution in `O(log(m * n))` time complexity.

Code

```
import java.util.*;
class Solution {
    public static boolean searchMatrix(int[][] matrix, int target) {
        ArrayList<Integer> fEle = new ArrayList<Integer>();

        for(int[] row : matrix){
            fEle.add(row[0]);
        }

        //binary search
        int low=0;
        int high=fEle.size()-1;
        int rowIndex = -1;
        while(low<=high){
            int mid = low + (high-low)/2;

            //break point
            if((mid==fEle.size()-1)||(target >= fEle.get(mid) && target <
fEle.get(mid+1))){
                rowIndex = mid;
                break;
            }

            if(target > fEle.get(mid)){
                low = mid+1;
            }
            else{
                high = mid-1;
            }

        }

        if(rowIndex==-1) return false;

        // actual row searching
        low=0;
        int[] searchRow = matrix[rowIndex];
        high=searchRow.length-1;
        while(low<=high){
```

```java
            int mid = low + (high-low)/2;

            if(target == searchRow[mid]){
                return true;
            }

            if(target > searchRow[mid]){
                low = mid+1;
            }
            else{
                high = mid-1;
            }
        }

        return false;
    }

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

        System.out.println("Enter number of rows:");
        int rows = sc.nextInt();
        System.out.println("Enter number of columns:");
        int cols = sc.nextInt();
        sc.nextLine(); // consume newline

        int[][] matrix = new int[rows][cols];
        System.out.println("Enter the matrix row by row (comma-separated
values):");

        for (int i = 0; i < rows; i++) {
            String line = sc.nextLine().trim();
            if (!line.matches("^[0-9,\\s-]+$")) {
                System.out.println("Invalid input format. Only numbers and commas
are allowed.");
                return;
            }

            String[] values = line.split(",");
            if (values.length != cols) {
                System.out.println("Invalid number of columns in row " + (i + 1));
                return;
            }

            for (int j = 0; j < cols; j++) {
                matrix[i][j] = Integer.parseInt(values[j].trim());
            }
        }

        System.out.println("Enter target value:");
        int target = sc.nextInt();

        Solution sol = new Solution();
        boolean found = searchMatrix(matrix, target);
        System.out.println("Result: " + (found ? "Target found!" : "Target not
found."));
    }
}
```

Terminal Output

```
$ java Solution
Enter number of rows:
3
Enter number of columns:
3
Enter the matrix row by row (comma-separated values):
1,5,6
6,8,7
11,25,10
Enter target value:
11
Result: Target found!
```

Problem : Right-Angled Triangle of Stars

Problem Statement

Write a program that prints a `right-angled triangle` made of stars (`*`) with **n rows**.

Each row `i` should contain exactly `i` stars.

Input

- An integer `n` representing the number of rows.

Output

- A right-angled triangle of stars with `n` rows.

Example

Input: `5`

Output:

```
*
**
***
****
*****
```

Code

```java
import java.util.Scanner;

class Solution{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows for the pattern.");

        int n = sc.nextInt();

        System.out.println();
        System.out.println("Pattern :");
        System.out.println();

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

Terminal Output

```
$ java Solution
Enter the number of rows for the pattern.
5

Pattern :

*
**
***
****
*****
```

Problem : # Inverted Right-Angled Triangle

Problem Statement

Write a program that prints an `inverted right-angled triangle` made of stars (`*`) with **n rows**.

The first row should contain `n` stars, the second row `n-1`, and so on, until the last row contains 1 star.

Input

- An integer `n` representing the number of rows.

Output

- An inverted right-angled triangle of stars with `n` rows.

Example

Input: `5`

Output:

```
*****
****
***
**
*
```

Code

```java
import java.util.Scanner;

class Solution{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows for the pattern.");

        int n = sc.nextInt();

        System.out.println();
        System.out.println("Pattern :");
        System.out.println();

        for (int i = n; i >= 1; i--) {
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Terminal Output

```
$ java Solution
Enter the number of rows for the pattern.
5

Pattern :

*****
****
***
**
*
```

Problem : Pyramid of Stars

Problem Statement

Write a program that prints a **symmetric pyramid** made of stars (`*`) with **n rows**.

- The first row should have 1 star centered.
- Each subsequent row should increase by 2 stars, forming a symmetric pyramid.

Input

- An integer `n` representing the number of rows.

Output

- A symmetric pyramid of stars with `n` rows.

Example

Input: `5`

Output:

```
    *
   ***
  *****
 *******
*********
```

Code

```java
import java.util.Scanner;

class Solution{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows for the pattern.");

        int n = sc.nextInt();

        System.out.println();
        System.out.println("Pattern :");
        System.out.println();

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n - i; j++) System.out.print(" ");
            for (int j = 1; j <= 2 * i - 1; j++) System.out.print("*");
            System.out.println();
        }
    }
}
```

Terminal Output

```
$ java Solution
Enter the number of rows for the pattern.
5

Pattern :

    *
   ***
  *****
 *******
```

Problem : Number Pyramid

Problem Statement

Write a program that prints a **pyramid of numbers** with `n` rows.

- Each row `i` should contain numbers from `1` up to `i`.
- The numbers should form a symmetric pyramid.

Input

- An integer `n` representing the number of rows.

Output

- A number pyramid with `n` rows.

Example

Input:

`5`

Output:

```
    1
   1 2
  1 2 3
 1 2 3 4
1 2 3 4 5
```

Code

```java
import java.util.Scanner;

class Solution{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows for the pattern.");

        int n = sc.nextInt();

        System.out.println();
        System.out.println("Pattern :");
        System.out.println();

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

Terminal Output

```
$ java Solution
Enter the number of rows for the pattern.
5

Pattern :

    1
   1 2
  1 2 3
 1 2 3 4
1 2 3 4 5
```

Problem : Floyd’s Triangle

Problem Statement

Write a program that prints **Floyd’s Triangle** — a right-angled triangle of **consecutive numbers** starting from `1`, arranged row by row.

- Row 1 contains 1 number, row 2 contains 2 numbers, row 3 contains 3 numbers, and so on.

Input

- An integer `n` representing the number of rows.

Output

- Floyd’s Triangle with `n` rows.

Example

Input:

`5`

Output:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Code

```java
import java.util.Scanner;

class Solution{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows for the pattern.");

        int n = sc.nextInt();

        System.out.println();
        System.out.println("Pattern :");
        System.out.println();

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

Terminal Output

```
$ java Solution
Enter the number of rows for the pattern.
5

Pattern :

1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Problem : Diamond Pattern

Problem Statement

Write a program that prints a **diamond pattern** of stars (`*`).

- The diamond has **n rows in the top half**.
- The bottom half mirrors the top half, forming a symmetric diamond.

Input

- An integer `n` representing the number of rows in the top half.

Output

- A symmetric diamond of stars.

Example

Input:

`5`

Output:

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

Code

```java
import java.util.Scanner;

class Solution{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows for the pattern.");

        int n = sc.nextInt();

        System.out.println();
        System.out.println("Pattern :");
        System.out.println();

        // Top half
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n - i; j++) System.out.print(" ");
            for (int j = 1; j <= 2 * i - 1; j++) System.out.print("*");
            System.out.println();
        }
        // Bottom half
        for (int i = n - 1; i >= 1; i--) {
            for (int j = 1; j <= n - i; j++) System.out.print(" ");
            for (int j = 1; j <= 2 * i - 1; j++) System.out.print("*");
            System.out.println();
        }
    }
}
```

Terminal Output

```
$ java Solution
Enter the number of rows for the pattern.
5

Pattern :

    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

Problem : Hollow Pyramid

Problem Statement

Write a program that prints a **hollow pyramid** made of stars (`*`) with **n rows**.

- Only the **edges** and the **base** of the pyramid should have stars; the inside should be empty (spaces).

Input

- An integer `n` representing the number of rows.

Output

- A hollow pyramid of stars with `n` rows.

Example

Input:

`5`

Output:

```
    *
   * *
  *   *
 *     *
*********
```

Code

```java
import java.util.Scanner;

class Solution{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows for the pattern.");

        int n = sc.nextInt();

        System.out.println();
        System.out.println("Pattern :");
        System.out.println();

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n - i; j++) System.out.print(" ");
            for (int j = 1; j <= 2 * i - 1; j++){
                if (j == 1 || j == 2 * i - 1 || i == n) System.out.print("*");
                else System.out.print(" ");
            }
            System.out.println();
        }
    }
}
```

Terminal Output

```
$ java Solution
Enter the number of rows for the pattern.
5

Pattern :

    *
   * *
  *   *
 *     *
*********
```

Problem : Pascal’s Triangle

Problem Statement

Write a program that prints the **first n rows of Pascal’s Triangle**.

- Each number in the triangle is the sum of the two numbers directly above it.
- The triangle starts with `1` at the top.

Input

- An integer `n` representing the number of rows.

Output

- The first `n` rows of Pascal’s Triangle.

Example

Input:

`5`

Output:

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

Code

```
import java.util.Scanner;

class Solution{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows for the pattern.");

        int n = sc.nextInt();

        System.out.println();
        System.out.println("Pattern :");
        System.out.println();

        for (int line = 0; line < n; line++) {
            int C = 1;  // c = curr line
            for (int i = 0; i < n - line - 1; i++) System.out.print(" ");
            for (int i = 0; i <= line; i++) {
                System.out.print(C + " ");
                C = C * (line - i) / (i + 1);
            }
            System.out.println();
        }
    }
}
```

Terminal Output

```
$ java Solution
Enter the number of rows for the pattern.
5

Pattern :

    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

Problem : Zig-Zag Pattern

Problem Statement

Write a program that prints a **zig-zag pattern of stars (`*`)** across 3 rows for `n` columns.

- The stars should follow a diagonal pattern, creating a zig-zag across the three rows.

Input

- An integer `n` representing the number of columns.

Output

- A 3-row zig-zag pattern of stars.

Example

Input:

`9`

Output:

```
  *   *   *
 * * * * *
*   *   *
```

Code

```java
import java.util.Scanner;

class Solution{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows for the pattern.");

        int n = sc.nextInt();

        System.out.println();
        System.out.println("Pattern :");
        System.out.println();

        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= n; j++) {
                if ((i + j) % 4 == 0 || (i == 2 && j % 4 == 0))
System.out.print("*");
                else System.out.print(" ");
            }
            System.out.println();
        }
    }
}
```

Terminal output

```
$ java Solution
Enter the number of rows for the pattern.
25

Pattern :

  *   *   *   *   *   *
 * * * * * * * * * * * *
*   *   *   *   *   *   *
```

Problem : Count Words, Lines, and Characters

Problem Statement

Write a program that reads a text file and prints the **total number of lines**, **words**, and **characters** in it.

Your program should handle any plain text file and produce accurate counts even if the file contains multiple spaces, blank lines, or special characters.

Input

A text file containing any number of lines, words, and characters.

Output

Print three counts:

- Total number of **lines**
- Total number of **words**
- Total number of **characters**

Example

Input File Content (`input.txt`):

```
Hello world
This is a test file.
It contains multiple lines.
```

Output

```
Lines: 3
Words: 9
Characters: 53
```

Code

```
import java.util.*;

class Solution{

    public static ArrayList<Integer> findDuplicateElements(int arr[]){
        ArrayList<Integer> result = new ArrayList<>();
        HashMap<Integer,Integer> map = new HashMap<>();
        for(int num : arr){
            map.put(num , map.getOrDefault(num,0)+1);
        }

        for(var pair : map.entrySet()){
            if(pair.getValue()!=1){
                result.add(pair.getKey());
            }
        }

        return result;
    }

    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        String inputRegex = "^\\s*(-?\\d{1,9})(\\s*,\\s*-?\\d{1,9})*\\s*$";

        System.out.println("Enter the integer array \n For eg.
1,2,6,3,56,45....");
        String inputArr = sc.nextLine();

        if(inputArr.matches(inputRegex)){
            System.out.println("The input array is corect.");
        }
        else{
            System.out.println("The input array is in-corect. Exiting the
program...");
            return;
        }

        int[] arr = Arrays.stream(inputArr.split(","))
                    .map(String::trim)
                    .mapToInt(Integer::parseInt)
                    .toArray();

        System.out.println("Input array "+Arrays.toString(arr));

        ArrayList<Integer> result = findDuplicateElements(arr);

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

Terminal Output

```
$ java Solution
Enter the file path. Please select only text files for output.
C:\Users\SDE\Desktop\JavaPractice\FileSystem\question_01\input.txt
Lines: 3
Words: 11
```

```
Characters: 58
End of Program.
```

Problem : Count Frequency of Each Word

Problem Statement

Write a program that reads a text file and prints how many times each **word** appears in it.

The program should:

- **Ignore case** (treat “Hello” and “hello” as the same word).
- **Ignore punctuation** (so “word,” and “word” are treated the same).

Input

- A text file (e.g., `input.txt`) containing any text.

Output

- A list of words with their frequency counts.

Example

Input File (`input.txt`):

```
Hello world!
This is a test. Hello again, world.
```

Expected Output:

```
hello: 2
world: 2
this: 1
is: 1
a: 1
test: 1
again: 1
```

Code

```java
import java.io.*;
import java.util.*;

public class Solution {
    public static void main(String[] args) throws IOException {

        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the file path. Please select only text files for
output.");
        String inputPath = sc.nextLine().trim();

        try{
            BufferedReader br = new BufferedReader(new FileReader(inputPath));
            Map<String, Integer> freq = new HashMap<>();
            String line;
            while ((line = br.readLine()) != null) {
                line = line.replaceAll("[^a-zA-Z ]", "").toLowerCase();
                for (String word : line.split("\\s+")) {
                    if (!word.isEmpty()) freq.put(word, freq.getOrDefault(word, 0)
+ 1);
                }
            }
            br.close();
            freq.forEach((k, v) -> System.out.println(k + " - " + v));
        }
        catch(Exception err){
            System.out.println("Found errors with the provided file-path.");
            System.out.println(err);
        }
        finally{
            System.out.println("Ending the Program.");
        }
    }
}
```

Terminal output

```
$ java Solution
Enter the file path. Please select only text files for output.
C:\Users\SDE\Desktop\JavaPractice\FileSystem\question_02\input.txt
a - 1
contains - 1
world - 1
file - 1
test - 1
this - 1
multiple - 1
is - 1
hello - 1
it - 1
lines - 1
Ending the Program.
```

Inside `input.txt`

```
Hello world
This is a test file.
It contains multiple lines.
```

Problem : Append Data to File

Problem Statement

Write a program that takes **user input** (such as names, scores, or any text) and **appends** it to an existing file, **without overwriting** the previous content.

If the file does not exist, the program should create it automatically before writing the data.

Input

- A text value entered by the user (e.g., a name, a score, or a sentence).
- A file (e.g., `data.txt`) where the input should be appended.

Output

- The input data is added to the end of the file while preserving all existing content.

Example

Initial File (`data.txt`):

```
Alice - 85
Bob - 90
```

User Input: `Charlie - 88`

Updated File (`data.txt`):

```
Alice - 85
Bob - 90
Charlie - 88
```

Code

```java
import java.io.*;
import java.util.Scanner;

public class Solution {
    public static void main(String[] args) throws IOException {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the string you want to append to a file.");
        String inputStr = sc.nextLine().trim();

        System.out.println("Enter the file path where you want to append. Please select only text files for output.");
        String inputPath = sc.nextLine().trim();

        try{
            FileWriter fw = new FileWriter(inputPath, true); // true → append mode
            fw.write(inputStr + "\n");
            fw.close();
            System.out.println("Data appended successfully.");
        }
        catch(Exception err){
            System.out.println("Found errors with the provided file-path.");
            System.out.println(err);
        }
        finally{
            System.out.println("Ending the Program.");
        }
    }
}
```

Terminal Output

```
$ java Solution
Enter the string you want to append to a file.
New data line
Enter the file path where you want to append. Please select only text files for output.
C:\Users\SDE\Desktop\JavaPractice\FileSystem\question_03\sample.txt
Data appended successfully.
Ending the Program.
```

Inside `sample.txt`

Before

```
Hello this is a line.
```

After

```
Hello this is a line.
New data line
```

Problem : Find and Replace in a File

Problem Statement

Write a program that reads the contents of a text file, replaces all occurrences of a **target word or phrase** with a **replacement word or phrase**, and writes the modified content to a new file.

Given:

- A file name (e.g., `input.txt`)
- Two strings:
  - **Find** → the text to search for
  - **Replace** → the text to replace it with

The program should create a new file (e.g., `output.txt`) containing the updated text.

Example

Input File (`input.txt`):

```
There was an error in the system.
The error caused a system crash.
```

Find: `error`
Replace: `warning`

Output File (`output.txt`):

```
There was a warning in the system.
The warning caused a system crash.
```

Code

```java
import java.io.*;
import java.nio.file.*;
import java.util.Scanner;

public class Solution {
    public static void main(String[] args) throws IOException {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the word that you want to find & replace");
        String findStr = sc.nextLine().trim();

        System.out.println("Enter the word that you want to replace with");
        String replaceStr = sc.nextLine().trim();

        System.out.println("Enter the file path. Please select only text files for output.");
        String inputPath = sc.nextLine().trim();

        try{
            String content = new String(Files.readAllBytes(Paths.get(inputPath)));
            content = content.replaceAll(findStr, replaceStr);
            Files.write(Paths.get("output.txt"), content.getBytes());
            System.out.println("Replacements done.");
        }
        catch(Exception err){
            System.out.println("Found errors with the provided file-path.");
            System.out.println(err);
        }
        finally{
            System.out.println("Ending the Program.");
        }
    }
}
```

Terminal Output

```
$ java Solution
Enter the word that you want to find & replace
system
Enter the word that you want to replace with
environment
Enter the file path. Please select only text files for output.
C:\Users\SDE\Desktop\JavaPractice\FileSystem\question_04\sample.txt
Replacements done.
Ending the Program.
```

Inside `sample.txt`

```
There was an error in the system.
The error caused a system crash.
```

Inside `output.txt`

```
There was an error in the environment.
The error caused a environment crash.
```

Problem : Copy Contents from One File to Another

Problem Statement

Write a program that reads the contents of `input.txt` and writes the **same contents** to a new file named `output.txt`.

If either file does not exist, your program should **create it automatically**.
Ensure that the copied file has the exact same text as the original, including spaces, newlines, and special characters.

Input

- A text file named `input.txt` containing any text.

Output

- A new text file named `output.txt` containing the same contents as `input.txt`.

Example

Input File (`input.txt`):

```
Hello World
This is a sample file.
File copy test.
```

Output File (`output.txt`):

```
Hello World
This is a sample file.
File copy test.
```

Code

```java
import java.io.*;
import java.util.Scanner;

public class Solution {
    public static void main(String[] args) throws IOException {

        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the file path. Select the source file.");
        String inputPath = sc.nextLine().trim();

        System.out.println("Enter the file path. Select the destination file.");
        String outputPath = sc.nextLine().trim();

        try{
            FileInputStream in = new FileInputStream(inputPath);
            FileOutputStream out = new FileOutputStream(outputPath);
            int c;
            while ((c = in.read()) != -1) {
                out.write(c);
            }
            in.close();
            out.close();
            System.out.println("File copied successfully.");
        }
        catch(Exception err){
            System.out.println("Found errors with the provided file-path.");
            System.out.println(err);
        }
        finally{
            System.out.println("Ending the Program.");
        }
    }
}
```

Terminal Output

```
$ java Solution
Enter the file path. Select the source file.
C:\Users\SDE\Desktop\JavaPractice\FileSystem\question_05\input.txt
Enter the file path. Select the destination file.
C:\Users\SDE\Desktop\JavaPractice\FileSystem\question_05\output.txt
File copied successfully.
Ending the Program.
```

Inside `input.txt`

```
Hello world
This is a test file.
It contains multiple lines.
```

Inside `output.txt`

```
Hello world
This is a test file.
It contains multiple lines.
```