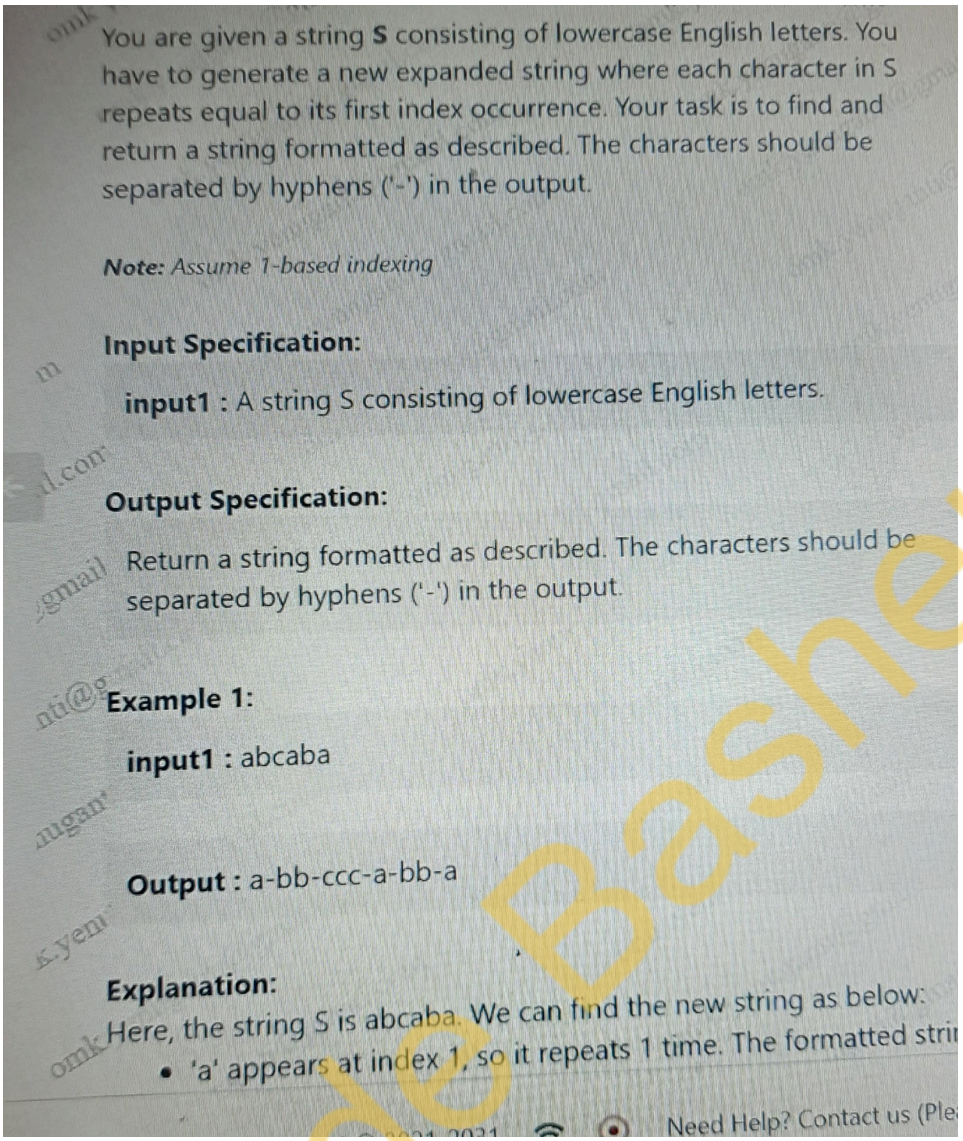
Q1.



You are given a string **S** consisting of lowercase English letters. You have to generate a new expanded string where each character in S repeats equal to its first index occurrence. Your task is to find and return a string formatted as described. The characters should be separated by hyphens ('-') in the output.

Note: *Assume 1-based indexing*

Input Specification:

input1 : A string S consisting of lowercase English letters.

Output Specification:

Return a string formatted as described. The characters should be separated by hyphens ('-') in the output.

Example 1:

input1 : abcaba

Output : a-bb-ccc-a-bb-a

Explanation:

Here, the string S is abcaba. We can find the new string as below:

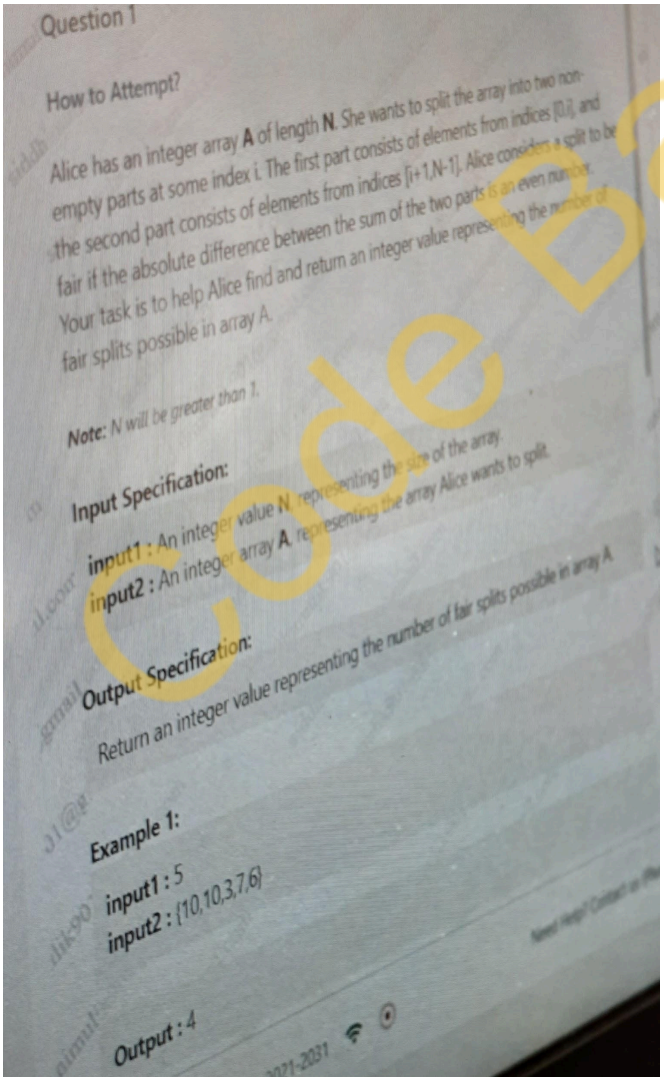
- 'a' appears at index 1, so it repeats 1 time. The formatted stri

Need Help? Contact us (Ple

```
public class ExpandedString {

    public static String expandString(String S) {
        StringBuilder result = new StringBuilder();
        int[] firstOccurrence = new int[26];

        for (int i = 0; i < S.length(); i++) {
            char ch = S.charAt(i);
            int index = ch - 'a';

            if (firstOccurrence[index] == 0) {
                firstOccurrence[index] = i + 1;
            }

            if (i > 0) {
                result.append("-");
            }
```

```
            int repeatCount = firstOccurrence[index];
            for (int j = 0; j < repeatCount; j++) {
                result.append(ch);
            }
        }

        return result.toString();
    }

    public static void main(String[] args) {
        String input = "abcaba";
        String output = expandString(input);
        System.out.println(output);
    }
}
```

Q2.



Question 1

How to Attempt?

Alice has an integer array **A** of length **N**. She wants to split the array into two non-empty parts at some index i. The first part consists of elements from indices [0,i], and the second part consists of elements from indices [i+1,N-1]. Alice considers a split to be fair if the absolute difference between the sum of the two parts is an even number. Your task is to help Alice find and return an integer value representing the number of fair splits possible in array A.

Note: *N will be greater than 1.*

Input Specification:

input1 : An integer value N, representing the size of the array.

input2 : An integer array **A**, representing the array Alice wants to split.

Output Specification:

Return an integer value representing the number of fair splits possible in array A.

Example 1:

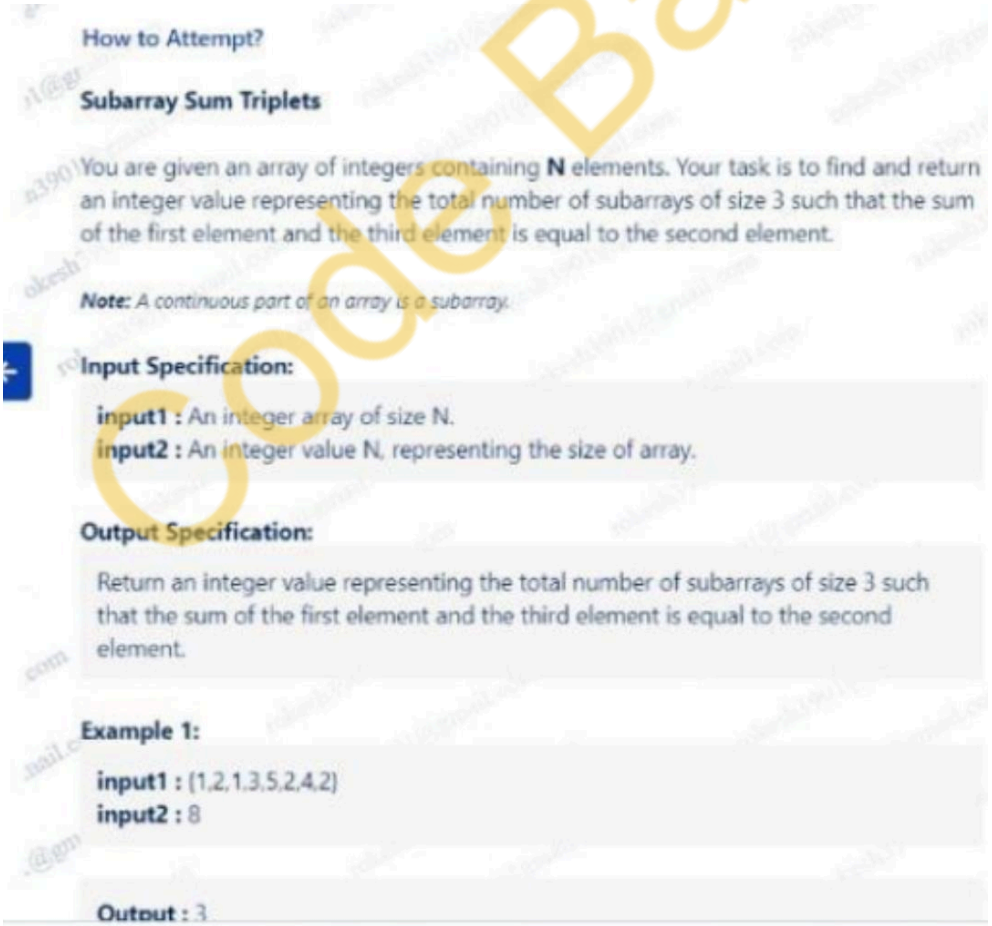
input1 : 5

input2 : {10,10,3,7,6}

Output : 4

```
public class FairSplits {
    public static int countFairSplits(int N, int[] A) {
        int totalSum = 0;
        for (int num : A) totalSum += num;
        int count = 0;
        int prefixSum = 0;
        for (int i = 0; i < N - 1; i++) {
            prefixSum += A[i];
            int suffixSum = totalSum - prefixSum;
            if (Math.abs(prefixSum - suffixSum) % 2 == 0) count++;
        }
        return count;
    }

    public static void main(String[] args) {
        int N = 5;
        int[] A = {10, 10, 3, 7, 6};
        System.out.println(countFairSplits(N, A));
    }
}
```

Q3.



How to Attempt?

Subarray Sum Triplets

You are given an array of integers containing **N** elements. Your task is to find and return an integer value representing the total number of subarrays of size 3 such that the sum of the first element and the third element is equal to the second element.

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

Input Specification:

input1 : An integer array of size N.
input2 : An integer value N, representing the size of array.

Output Specification:

Return an integer value representing the total number of subarrays of size 3 such that the sum of the first element and the third element is equal to the second element.

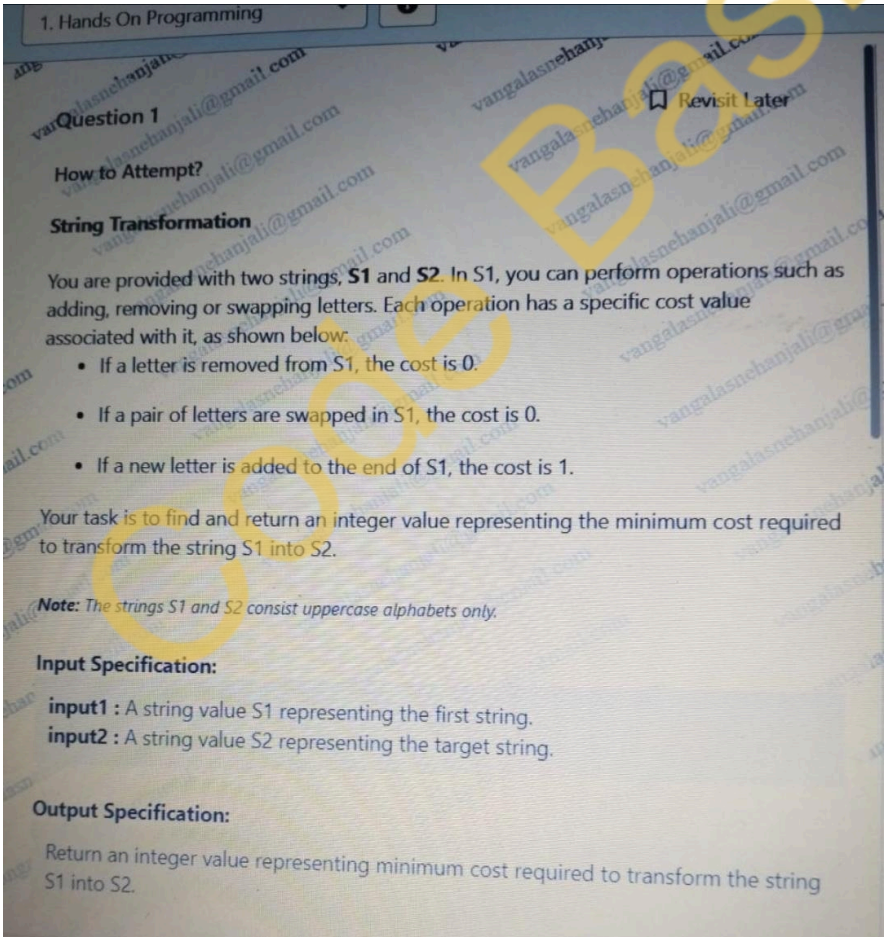
Example 1:

input1 : {1,2,1,3,5,2,4,2}
input2 : 8

Output : 3

```
public class SubarraySumTriplets {
    public static int countTriplets(int[] arr, int N) {
        int count = 0;
        for (int i = 0; i <= N - 3; i++) {
            if (arr[i] + arr[i + 2] == arr[i + 1]) {
                count++;
            }
        }
        return count;
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 1, 3, 5, 2, 4, 2};
        int N = 8;
        System.out.println(countTriplets(arr, N));
    }
}
```

Q4.



1. Hands On Programming

varQuestion 1

Revisit Later

How to Attempt?

String Transformation

You are provided with two strings, **S1** and **S2**. In S1, you can perform operations such as adding, removing or swapping letters. Each operation has a specific cost value associated with it, as shown below:

- If a letter is removed from S1, the cost is 0.
- If a pair of letters are swapped in S1, the cost is 0.
- If a new letter is added to the end of S1, the cost is 1.

Your task is to find and return an integer value representing the minimum cost required to transform the string S1 into S2.

Note: *The strings S1 and S2 consist uppercase alphabets only.*

Input Specification:

input1 : A string value S1 representing the first string.
input2 : A string value S2 representing the target string.

Output Specification:

Return an integer value representing minimum cost required to transform the string S1 into S2.

Output Specification:

Return an integer value representing minimum cost required to transform the string S1 into S2.

Example 1:

input1 : ABD
input2 : AABCCAD

Output : 4

Explanation:
Here, S1 = ABD and S2 = AABCCAD. We can perform the following steps to convert S1 into S2:

- Add a new element 'A' at the end of S1. This will make the composition ABDA. Now swap the last 'A', first with 'D' and then with 'B'. This will result in AABD and the cost will be 1.
- Add two 'C's at the end. This will make the composition AABDCC. After this, swap the 'D' with the last 'C'. This will result in AABCCD and the cost will be 2.
- Add one 'A' to AABCCD to make the string AABCCDA and then swap the last 'A' with 'D'. This will finally make the string AABCCAD, and the cost will be 1.

The total cost will be 4 (1 + 2 + 1). Hence, **4** is returned as the output.

into S2:

- Add a new element 'A' at the end of S1. This will make the composition ABDA. Now swap the last 'A', first with 'D' and then with 'B'. This will result in AABD and the cost will be 1.
- Add two 'C's at the end. This will make the composition AABDCC. After this, swap the 'D' with the last 'C'. This will result in AABCCD and the cost will be 2.
- Add one 'A' to AABCCD to make the string AABCCDA and then swap the last 'A' with 'D'. This will finally make the string AABCCAD, and the cost will be 1.

The total cost will be 4 (1 + 2 + 1). Hence, **4** is returned as the output.

Example 2:

input1 : ABC
input2 : XYZ

Output : 3

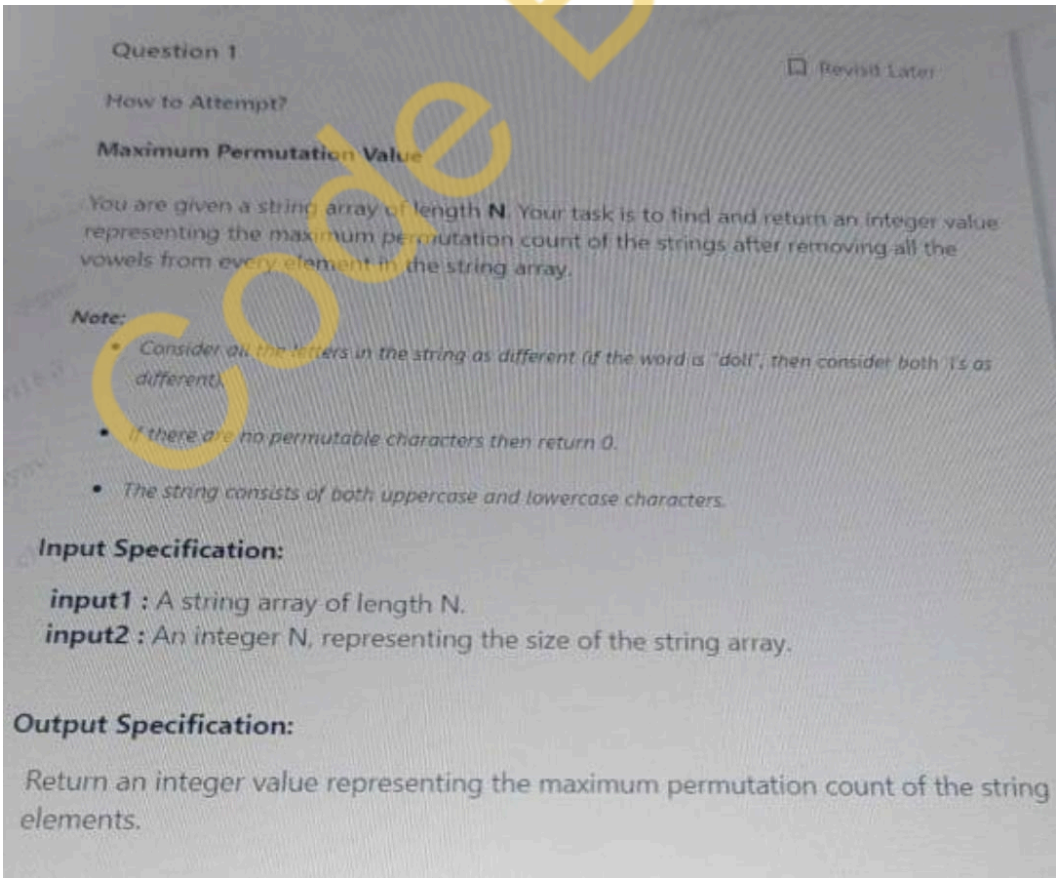
Explanation:
Here, S1 = ABC and S2 = XYZ. We can perform the following steps to convert S1 into S2:

- Remove all the elements from S1. This will make it an empty string, which will cost 0.
- Add X, Y and then Z, which will cost 3.

Since the total cost is 3, **3** is returned as the output.

```
public class TransformString {
    public static int minCost(String S1, String S2) {
        int[] freq1 = new int[26];
        int[] freq2 = new int[26];
        for (char c : S1.toCharArray()) freq1[c - 'A']++;
        for (char c : S2.toCharArray()) freq2[c - 'A']++;
        int cost = 0;
        for (int i = 0; i < 26; i++) {
            if (freq2[i] > freq1[i]) {
                cost += freq2[i] - freq1[i];
            }
        }
        return cost;
    }

    public static void main(String[] args) {
        String S1 = "ABD";
        String S2 = "AABCCAD";
        System.out.println(minCost(S1, S2));
    }
}
```

Q5.



Question 1

Revisit Later

How to Attempt?

Maximum Permutation Value

You are given a string array of length **N**. Your task is to find and return an integer value representing the maximum permutation count of the strings after removing all the vowels from every element in the string array.

Note:

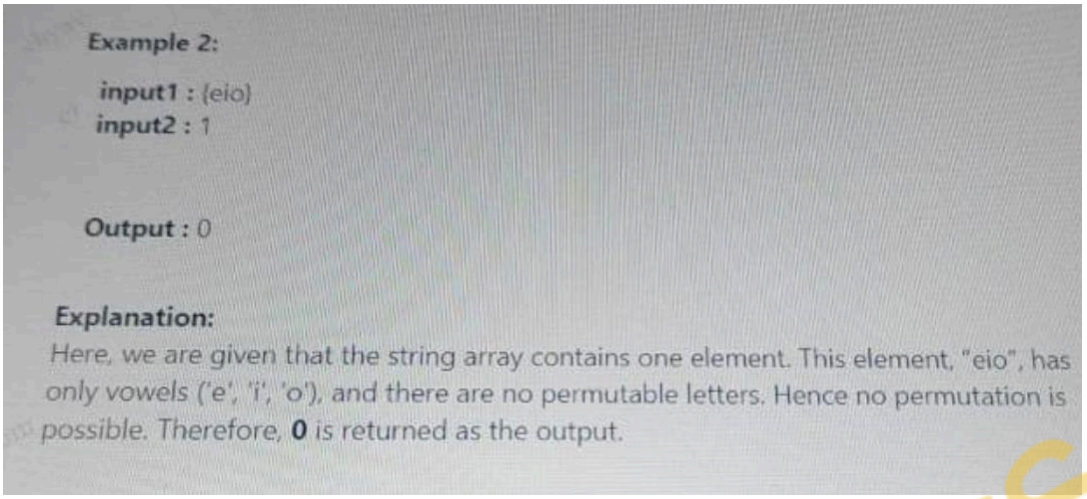
- *Consider all the letters in the string as different (if the word is "doll", then consider both 'l's as different).*
- *If there are no permutable characters then return 0.*
- *The string consists of both uppercase and lowercase characters.*

Input Specification:

input1 : *A string array of length N.*
input2 : *An integer N, representing the size of the string array.*

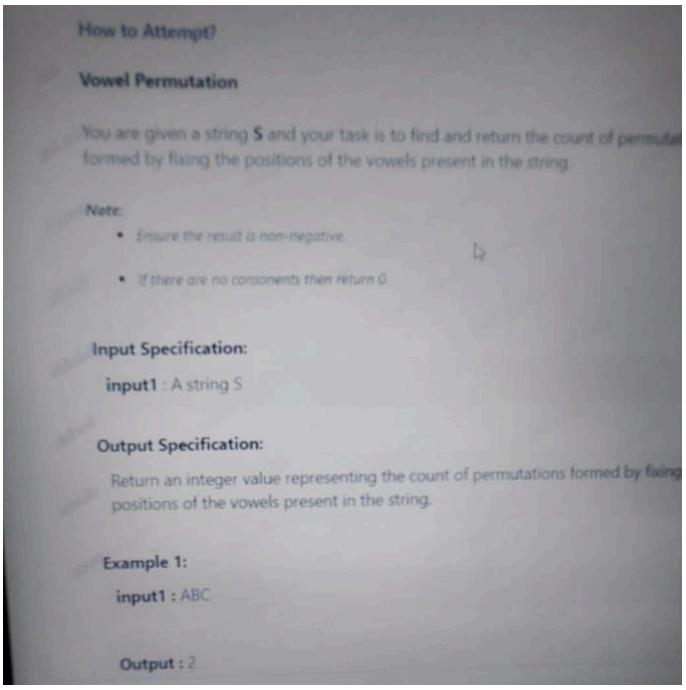
Output Specification:

Return an integer value representing the maximum permutation count of the string elements.




```
import java.util.*;

public class MaxPermutationValue {
    public static long factorial(int n) {
        long fact = 1;
        for (int i = 2; i <= n; i++) fact *= i;
        return fact;
    }

    public static long maxPermutation(String[] arr, int N) {
        int maxLen = 0;
        for (String s : arr) {
            String noVowels = s.replaceAll("(?i)[aeiou]", "");
            maxLen = Math.max(maxLen, noVowels.length());
        }
        return maxLen == 0 ? 0 : factorial(maxLen);
    }

    public static void main(String[] args) {
        String[] arr = {"eio"};
        int N = 1;
        System.out.println(maxPermutation(arr, N));
    }
}
```

Q6.



How to Attempt?

Vowel Permutation

You are given a string **S** and your task is to find and return the count of permuta
formed by fixing the positions of the vowels present in the string.

Note:

- *Ensure the result is non-negative.*
- *If there are no consonents then return 0.*

Input Specification:

input1 : A string S

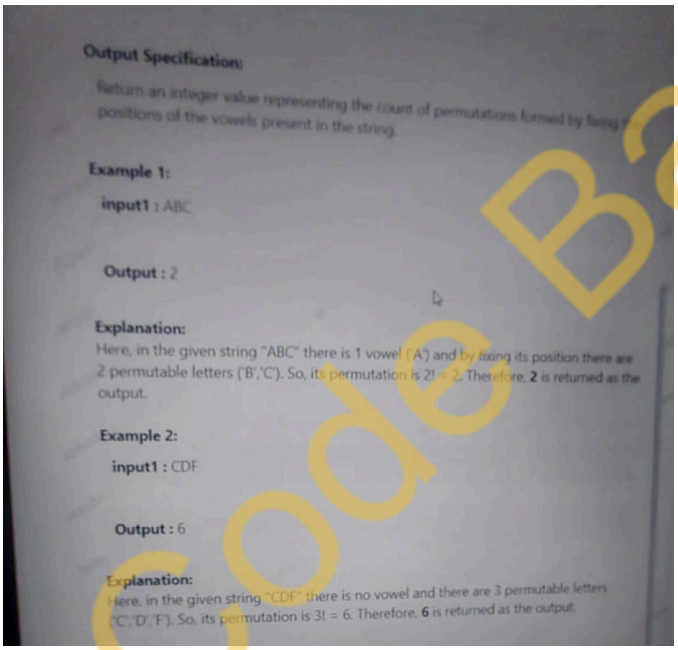
Output Specification:

Return an integer value representing the count of permutations formed by fixing
positions of the vowels present in the string.

Example 1:

input1 : ABC

Output : 2



Output Specification:

Return an integer value representing the count of permutations formed by fixing t
positions of the vowels present in the string.

Example 1:

input1 : ABC

Output : 2

Explanation:
Here, in the given string "ABC" there is 1 vowel ('A') and by fixing its position there are
2 permutable letters ('B','C'). So, its permutation is 2! = 2. Therefore, **2** is returned as the
output.

Example 2:

input1 : CDF

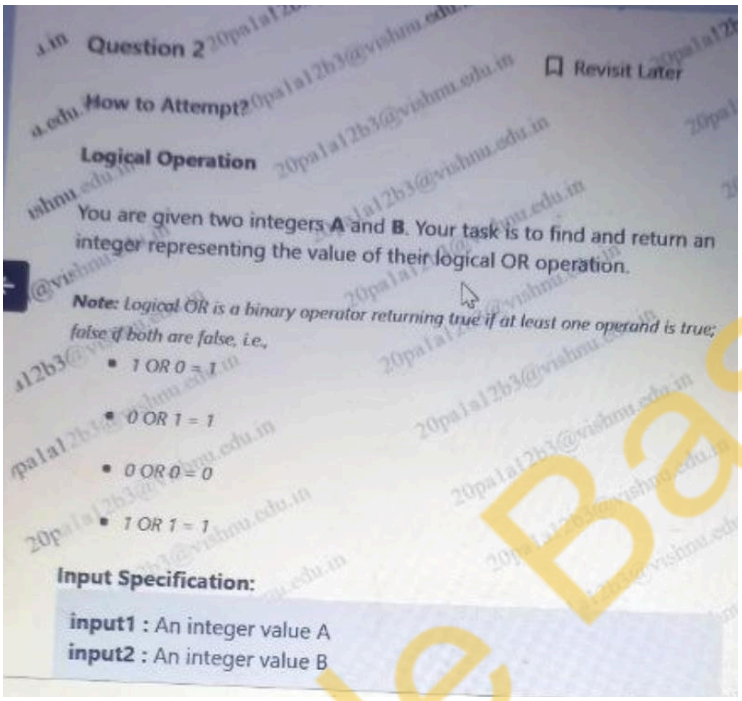
Output : 6

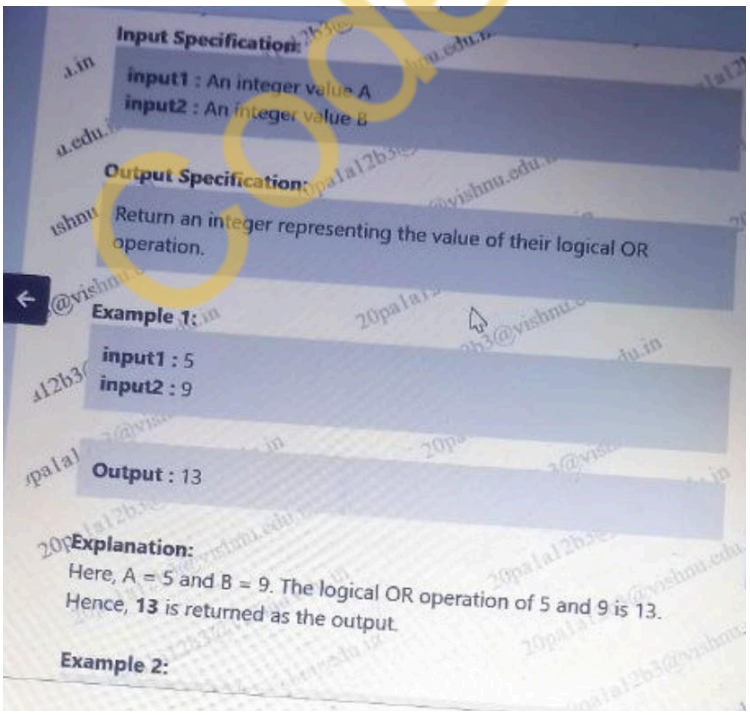
Explanation:
Here, in the given string "CDF" there is no vowel and there are 3 permutable letters
('C','D','F'). So, its permutation is 3! = 6. Therefore, **6** is returned as the output.

```
public class VowelPermutation {
    public static long factorial(int n) {
        long fact = 1;
        for (int i = 2; i <= n; i++) fact *= i;
        return fact;
    }

    public static long countPermutations(String S) {
        int consonants = 0;
        for (char c : S.toCharArray()) {
            if (!"AEIOUaeiou".contains(String.valueOf(c))) consonants++;
        }
```

```
        return consonants == 0 ? 0 : factorial(consonants);
    }

    public static void main(String[] args) {
        String S = "ABC";
        System.out.println(countPermutations(S));
    }
}
```

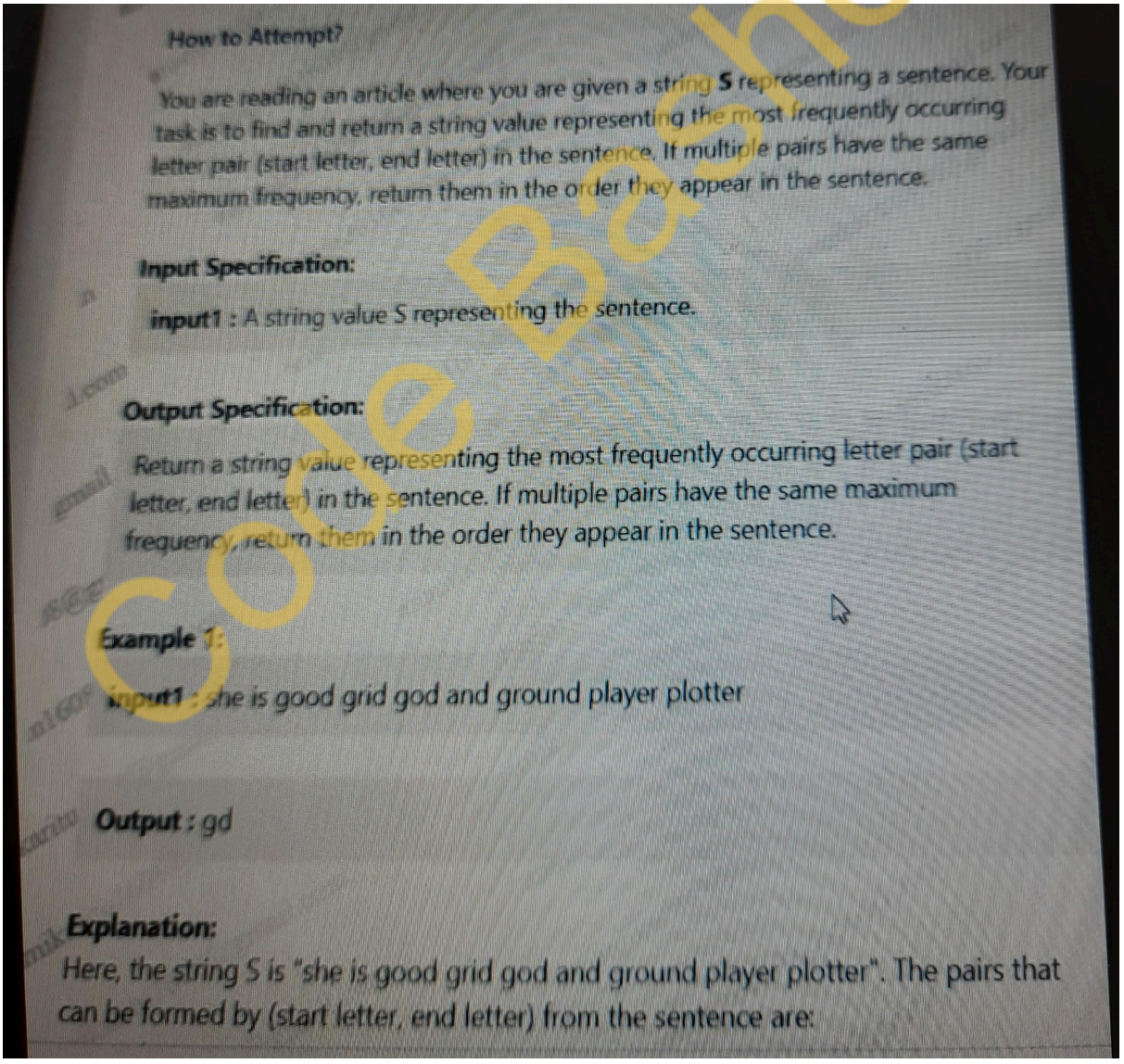
Q7.





```
import java.util.*;

public class LogicalOR {
    public static int logicalOr(int A, int B) {
        return A | B;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int A = sc.nextInt();
        int B = sc.nextInt();
        System.out.println(logicalOr(A, B));
    }
}
```

Q8.



How to Attempt?

You are reading an article where you are given a string **S** representing a sentence. Your task is to find and return a string value representing the most frequently occurring letter pair (start letter, end letter) in the sentence. If multiple pairs have the same maximum frequency, return them in the order they appear in the sentence.

Input Specification:

input1 : A string value S representing the sentence.

Output Specification:

Return a string value representing the most frequently occurring letter pair (start letter, end letter) in the sentence. If multiple pairs have the same maximum frequency, return them in the order they appear in the sentence.

Example 1:

input1 : she is good grid god and ground player plotter

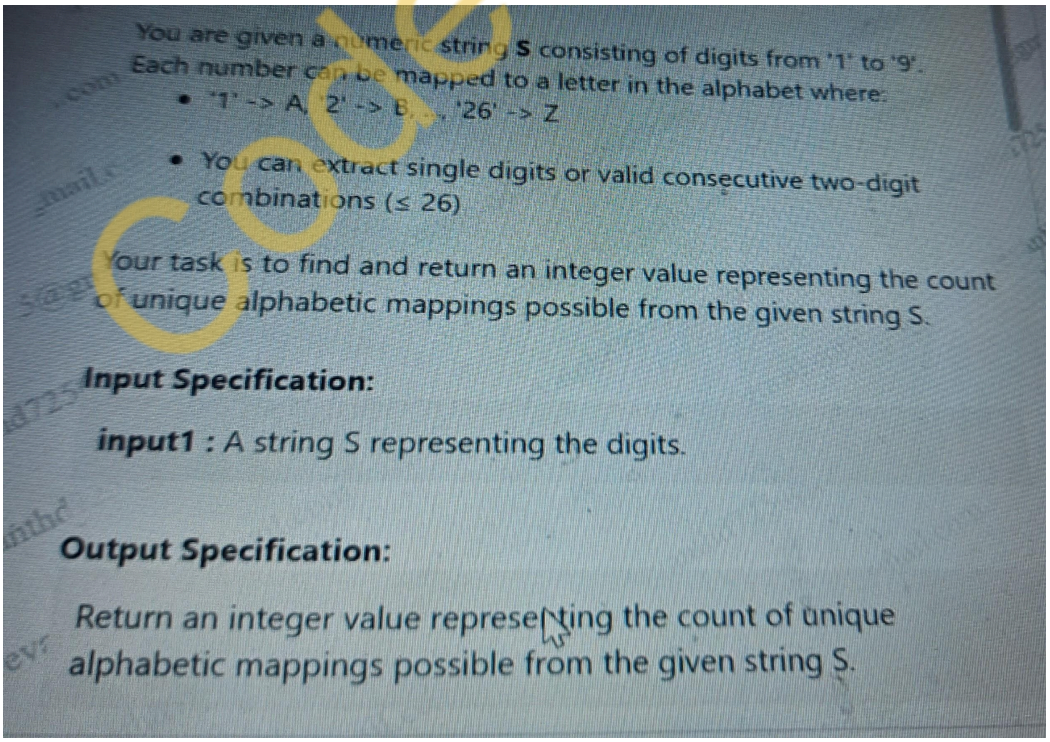
Output : gd

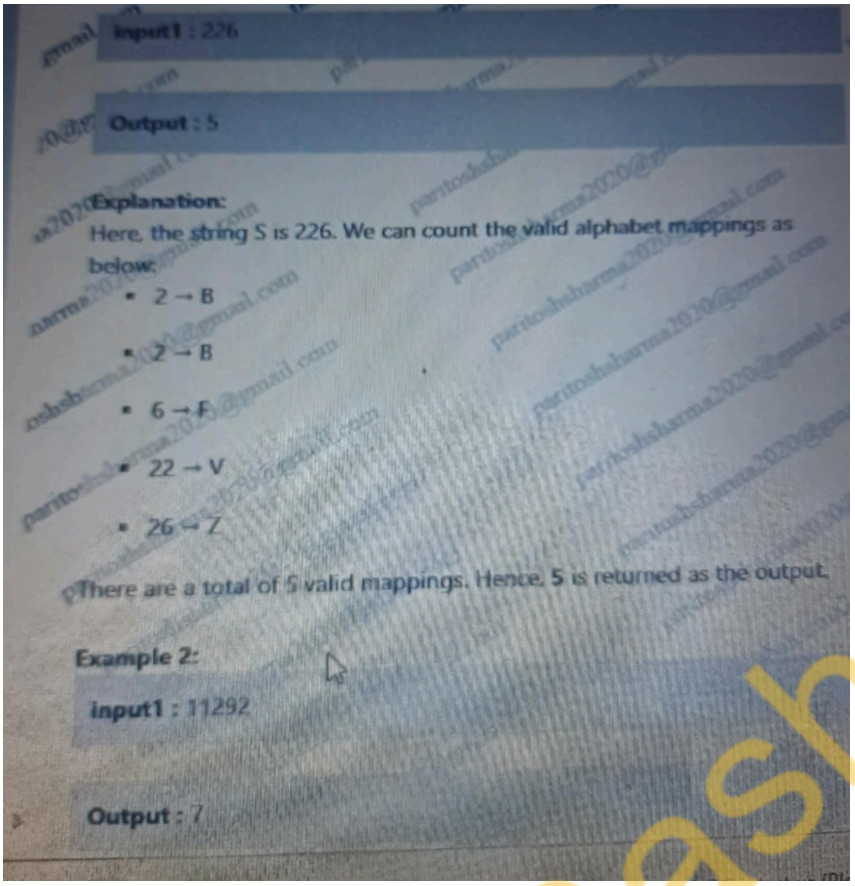
Explanation:

Here, the string S is "she is good grid god and ground player plotter". The pairs that can be formed by (start letter, end letter) from the sentence are:

```
import java.util.*;

public class MostFrequentPair {
    public static String mostFrequentPair(String S) {
        String[] words = S.split("\\s+");
        Map<String, Integer> freq = new LinkedHashMap<>();
        for (String word : words) {
            if (word.length() > 0) {
                String pair = "" + word.charAt(0) + word.charAt(word.length() - 1);
                freq.put(pair, freq.getOrDefault(pair, 0) + 1);
            }
        }
        int maxFreq = Collections.max(freq.values());
        for (String key : freq.keySet()) {
            if (freq.get(key) == maxFreq) return key;
        }
        return "";
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String S = sc.nextLine();
        System.out.println(mostFrequentPair(S));
    }
}
```

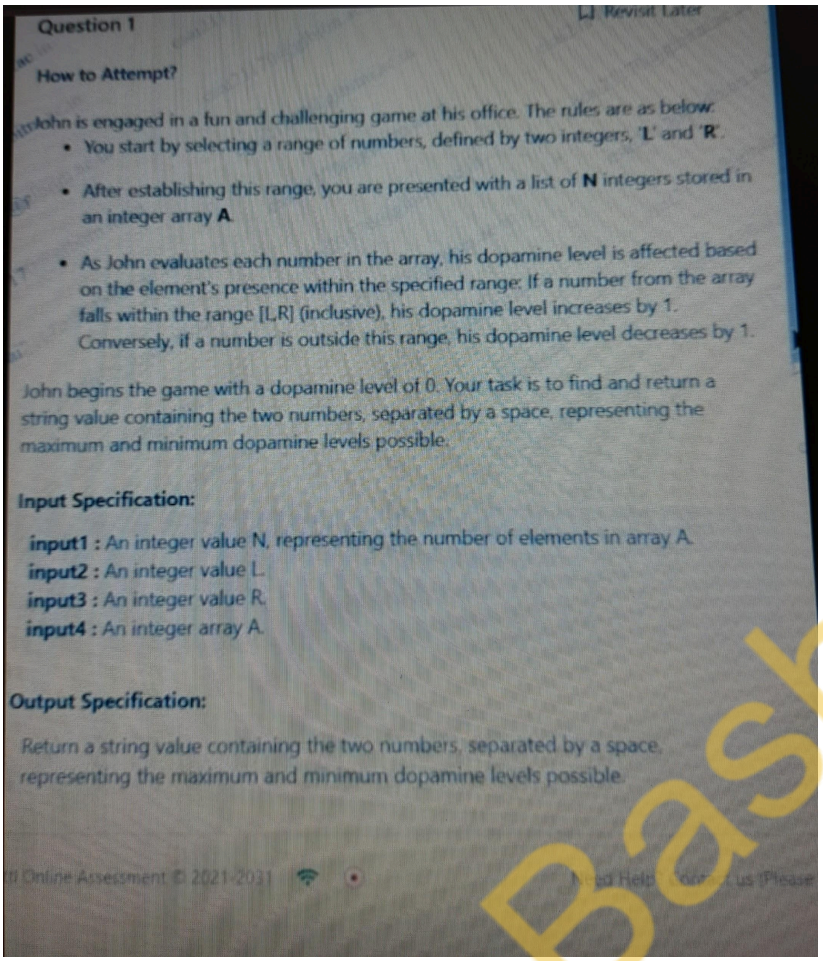
Q9.






```
import java.util.*;

public class UniqueMappings {
    public static int countMappings(String S) {
        int count = 0;
        for (int i = 0; i < S.length(); i++) {
            int num1 = S.charAt(i) - '0';
            if (num1 >= 1 && num1 <= 26) count++;
            if (i + 1 < S.length()) {
                int num2 = Integer.parseInt(S.substring(i, i + 2));
                if (num2 >= 1 && num2 <= 26) count++;
            }
        }
        return count;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String S = sc.nextLine();
        System.out.println(countMappings(S));
    }
}
```

Q10.



Question 1

Revisit Later

How to Attempt?

John is engaged in a fun and challenging game at his office. The rules are as below:

- You start by selecting a range of numbers, defined by two integers, '**L**' and '**R**'.
- After establishing this range, you are presented with a list of **N** integers stored in an integer array **A**.
- As John evaluates each number in the array, his dopamine level is affected based on the element's presence within the specified range: If a number from the array falls within the range [L,R] (inclusive), his dopamine level increases by 1. Conversely, if a number is outside this range, his dopamine level decreases by 1.

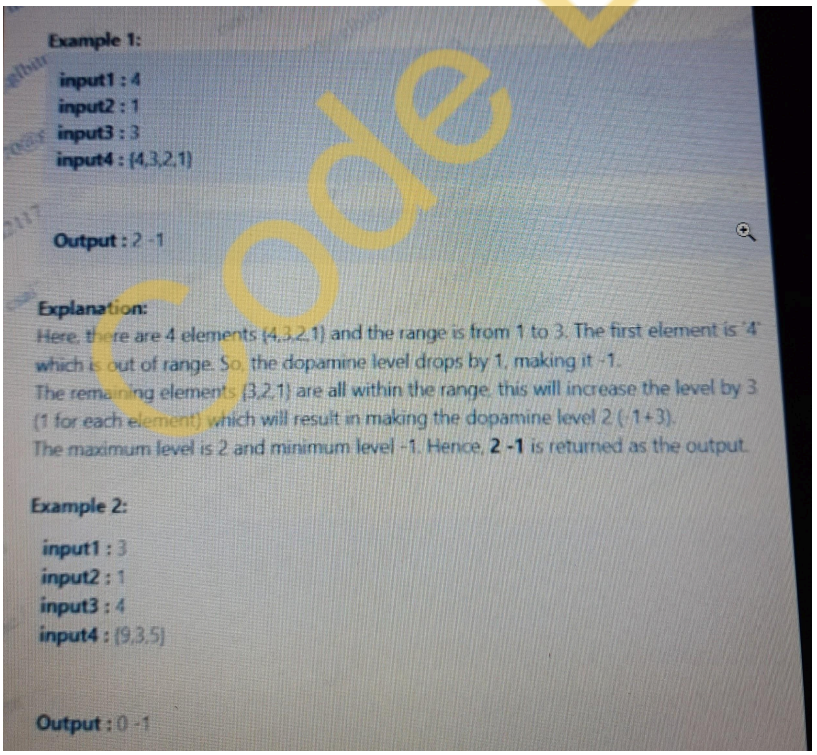
John begins the game with a dopamine level of 0. Your task is to find and return a string value containing the two numbers, separated by a space, representing the maximum and minimum dopamine levels possible.

Input Specification:

input1 : An integer value N, representing the number of elements in array A.
input2 : An integer value L.
input3 : An integer value R.
input4 : An integer array A.

Output Specification:

Return a string value containing the two numbers, separated by a space, representing the maximum and minimum dopamine levels possible.



Example 1:

input1 : 4
input2 : 1
input3 : 3
input4 : {4,3,2,1}

Output : 2 -1

Explanation:
Here, there are 4 elements {4,3,2,1} and the range is from 1 to 3. The first element is '4' which is out of range. So, the dopamine level drops by 1, making it -1.
The remaining elements {3,2,1} are all within the range, this will increase the level by 3 (1 for each element) which will result in making the dopamine level 2 (-1+3).
The maximum level is 2 and minimum level -1. Hence, **2 -1** is returned as the output.

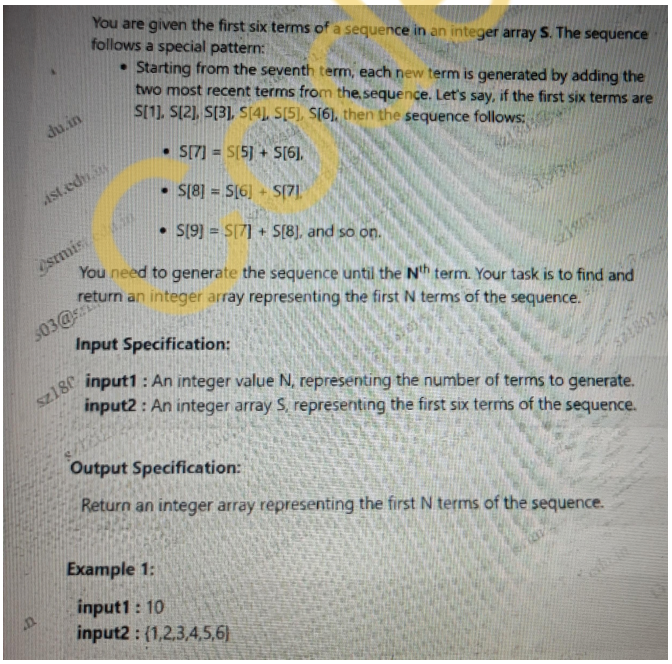
Example 2:

input1 : 3
input2 : 1
input3 : 4
input4 : {9,3,5}

Output : 0 -1

```java
import java.util.*;

public class DopamineLevels {
    public static String findDopamineLevels(int N, int L, int R, int[] A) {
        int dopamine = 0;
        int maxDopamine = Integer.MIN_VALUE;
        int minDopamine = Integer.MAX_VALUE;
        for (int num : A) {
            if (num >= L && num <= R) dopamine++;
            else dopamine--;
            maxDopamine = Math.max(maxDopamine, dopamine);
            minDopamine = Math.min(minDopamine, dopamine);
        }
        return maxDopamine + " " + minDopamine;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int L = sc.nextInt();
        int R = sc.nextInt();
        int[] A = new int[N];
        for (int i = 0; i < N; i++) A[i] = sc.nextInt();
        System.out.println(findDopamineLevels(N, L, R, A));
    }
}
```

Q11.



You are given the first six terms of a sequence in an integer array **S**. The sequence follows a special pattern:

- Starting from the seventh term, each new term is generated by adding the two most recent terms from the sequence. Let's say, if the first six terms are S[1], S[2], S[3], S[4], S[5], S[6], then the sequence follows:
  - S[7] = S[5] + S[6],
  - S[8] = S[6] + S[7],
  - S[9] = S[7] + S[8], and so on.

You need to generate the sequence until the **N**th term. Your task is to find and return an integer array representing the first N terms of the sequence.

Input Specification:

input1 : An integer value N, representing the number of terms to generate.
input2 : An integer array S, representing the first six terms of the sequence.

Output Specification:

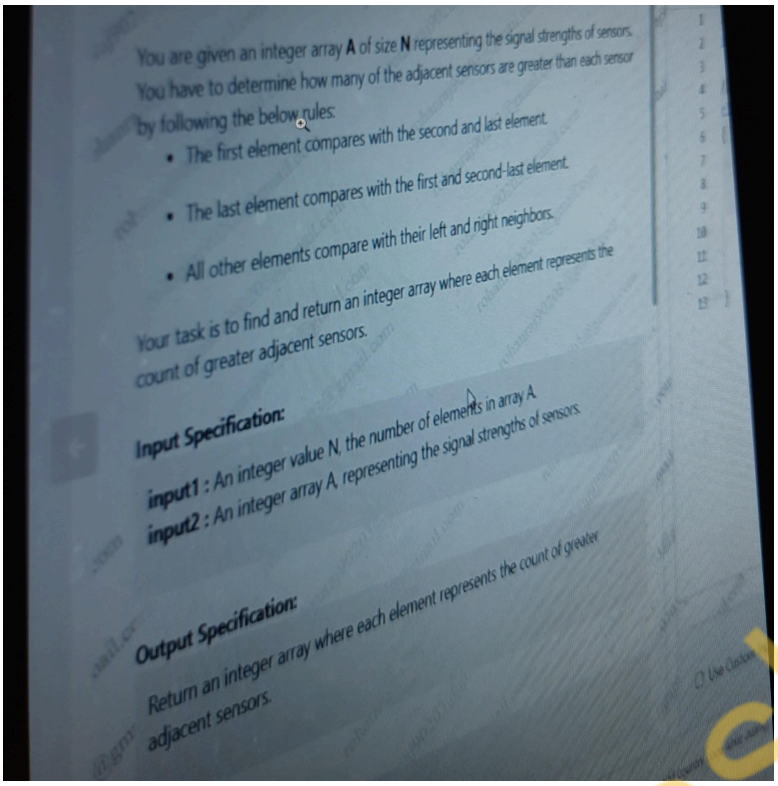
Return an integer array representing the first N terms of the sequence.

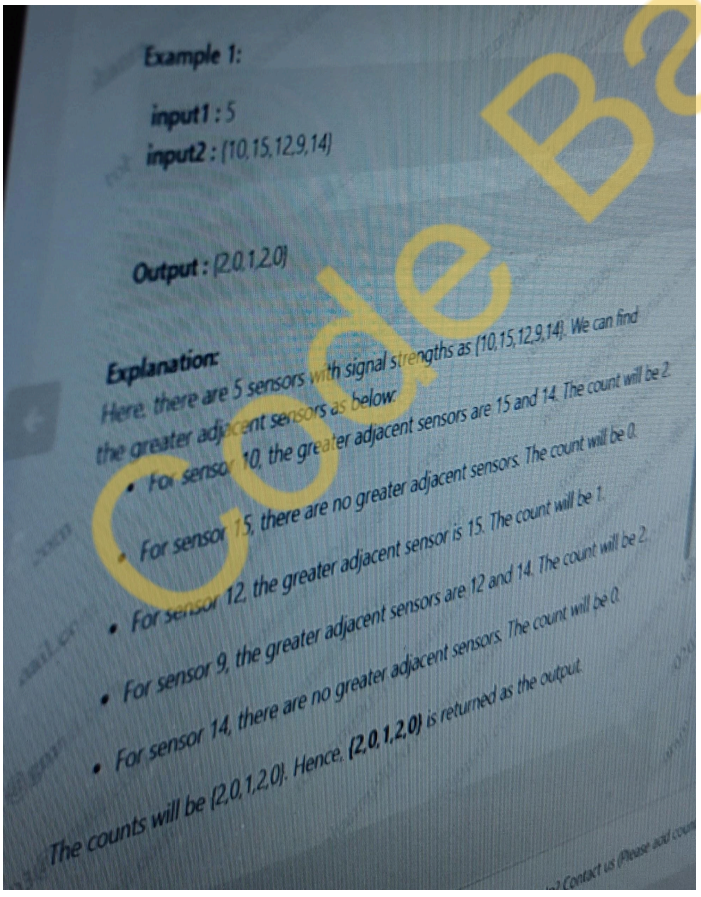
Example 1:

input1 : 10
input2 : {1,2,3,4,5,6}

```java
import java.util.*;

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

        int N = sc.nextInt();
        int[] S = new int[6];
        for (int i = 0; i < 6; i++) {
            S[i] = sc.nextInt();
        }

        int[] result = generateSequence(N, S);

        for (int i = 0; i < N; i++) {
            System.out.print(result[i] + " ");
        }
    }

    public static int[] generateSequence(int N, int[] S) {
        int[] seq = new int[N];

        for (int i = 0; i < Math.min(N, 6); i++) {
            seq[i] = S[i];
        }

        for (int i = 6; i < N; i++) {
            seq[i] = seq[i - 2] + seq[i - 1];
        }

        return seq;
    }
}
```

Q12.

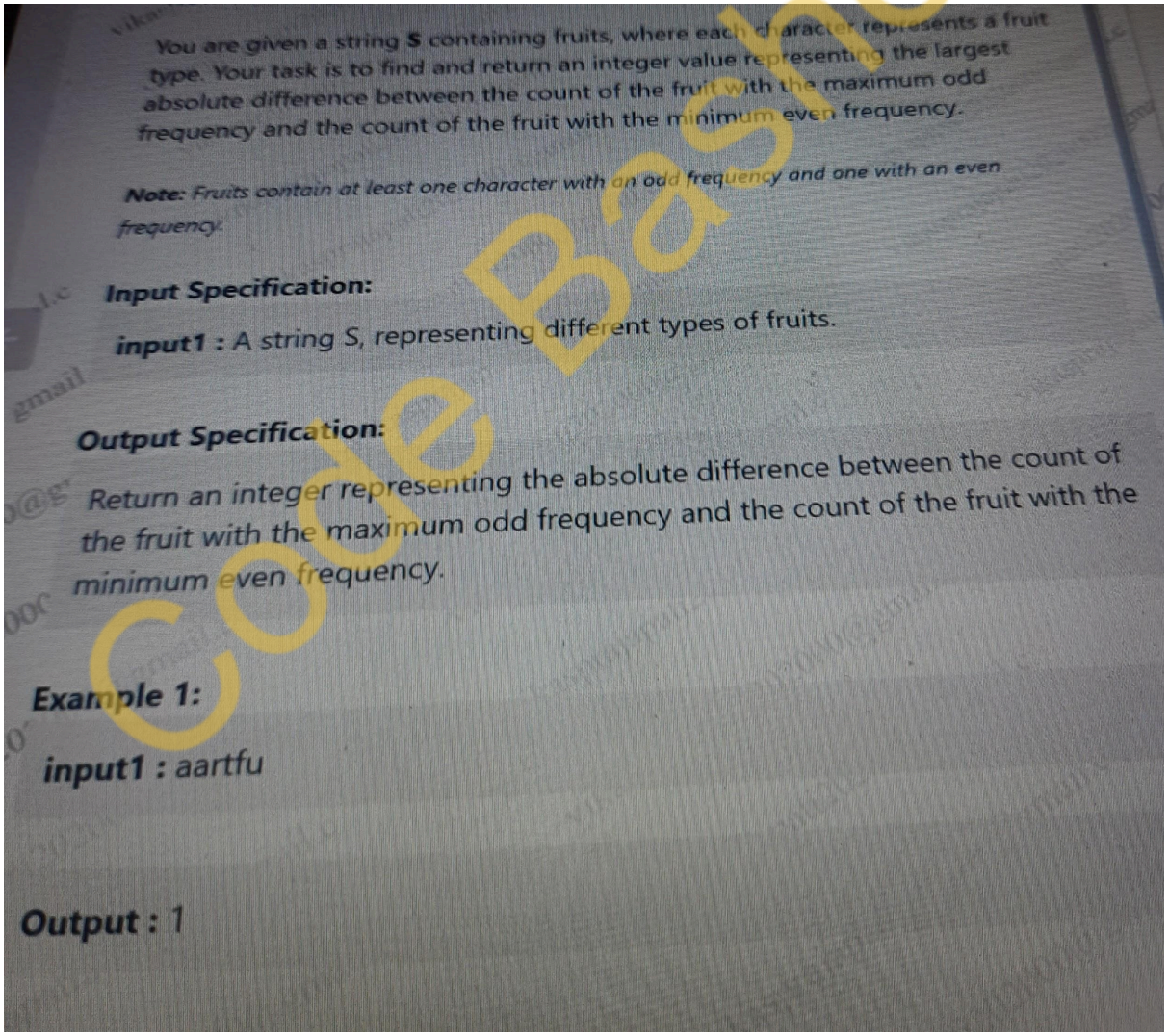





```
import java.util.*;

public class Main {
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int[] A = new int[N];
        for (int i = 0; i < N; i++) A[i] = sc.nextInt();
        for (int i = 0; i < N; i++) {
            int count = 0;
            if (A[(i - 1 + N) % N] > A[i]) count++;
            if (A[(i + 1) % N] > A[i]) count++;
            System.out.print(count);
            if (i < N - 1) System.out.print(" ");
        }
    }
}
```

Q13.



You are given a string **S** containing fruits, where each character represents a fruit type. Your task is to find and return an integer value representing the largest absolute difference between the count of the fruit with the maximum odd frequency and the count of the fruit with the minimum even frequency.

Note: *Fruits contain at least one character with an odd frequency and one with an even frequency.*

Input Specification:

input1 : A string S, representing different types of fruits.

Output Specification:

Return an integer representing the absolute difference between the count of the fruit with the maximum odd frequency and the count of the fruit with the minimum even frequency.

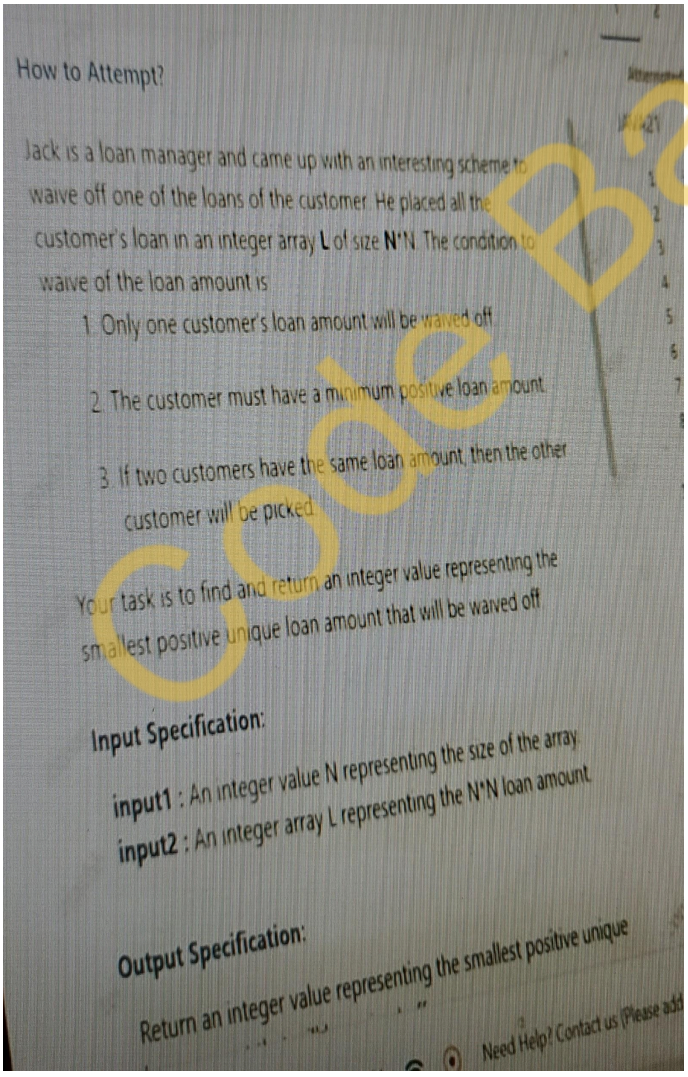
Example 1:

input1 : aartfu

Output : 1

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String S = sc.nextLine();
        Map<Character, Integer> freq = new HashMap<>();
        for (char c : S.toCharArray()) freq.put(c, freq.getOrDefault(c, 0) + 1);
        int maxOdd = Integer.MIN_VALUE;
        int minEven = Integer.MAX_VALUE;
        for (int count : freq.values()) {
            if (count % 2 == 1) maxOdd = Math.max(maxOdd, count);
            else minEven = Math.min(minEven, count);
        }
        System.out.println(Math.abs(maxOdd - minEven));
    }
}
```

Q14.



How to Attempt?

Jack is a loan manager and came up with an interesting scheme to waive off one of the loans of the customer. He placed all the customer's loan in an integer array **L** of size **N*N**. The condition to waive of the loan amount is

1. Only one customer's loan amount will be waived off.
2. The customer must have a minimum positive loan amount.
3. If two customers have the same loan amount, then the other customer will be picked.

Your task is to find and return an integer value representing the smallest positive unique loan amount that will be waived off.

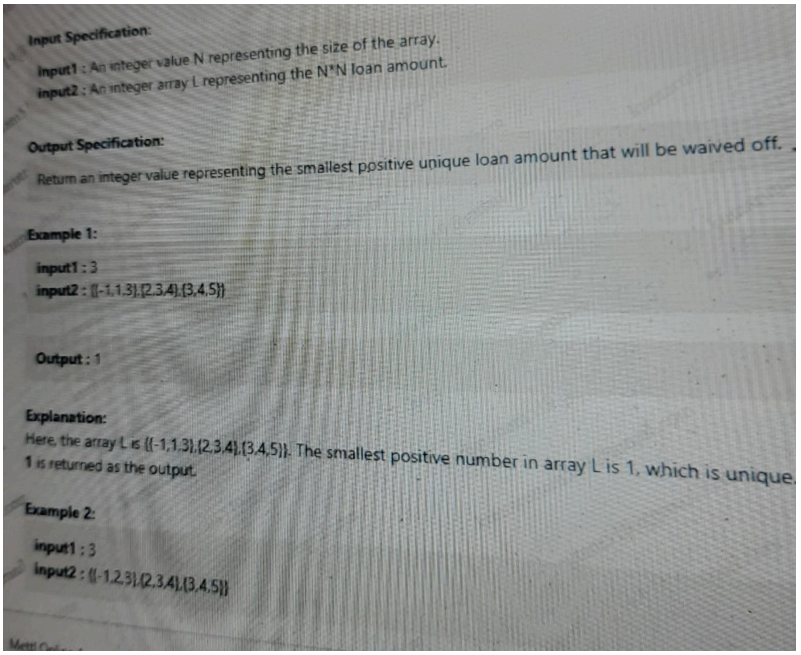
Input Specification:

input1 : An integer value N representing the size of the array.
input2 : An integer array L representing the N*N loan amount.

Output Specification:

Return an integer value representing the smallest positive unique

Need Help? Contact us (Please add



Input Specification:

input1 : An integer value N representing the size of the array.
input2 : An integer array L representing the N*N loan amount.

Output Specification:

Return an integer value representing the smallest positive unique loan amount that will be waived off.

Example 1:

input1 : 3
input2 : {{-1,1,3},{2,3,4},{3,4,5}}

Output : 1

Explanation:

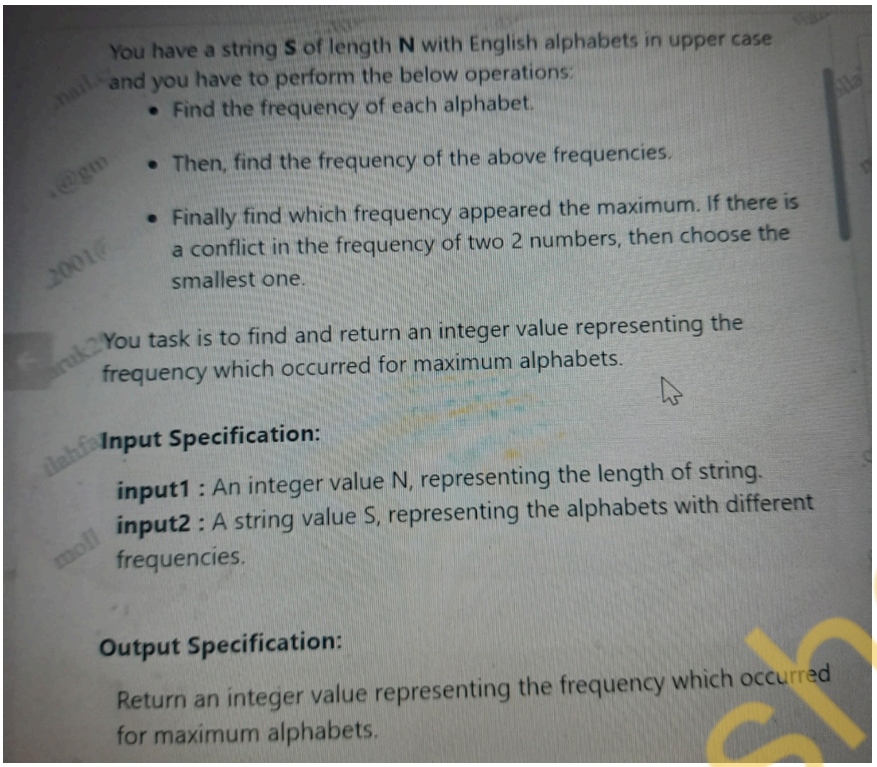
Here, the array L is {{-1,1,3},{2,3,4},{3,4,5}}. The smallest positive number in array L is 1, which is unique. 1 is returned as the output.

Example 2:

input1 : 3
input2 : {{-1,2,3},{2,3,4},{3,4,5}}

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int[] L = new int[N];
        for (int i = 0; i < N; i++) L[i] = sc.nextInt();
        Map<Integer, Integer> freq = new HashMap<>();
        for (int x : L) freq.put(x, freq.getOrDefault(x, 0) + 1);
        int ans = Integer.MAX_VALUE;
        for (int key : freq.keySet()) {
            if (key > 0 && freq.get(key) == 1) ans = Math.min(ans, key);
        }
        System.out.println(ans == Integer.MAX_VALUE ? -1 : ans);
    }
}
```

Q15.



You have a string **S** of length **N** with English alphabets in upper case and you have to perform the below operations:

- Find the frequency of each alphabet.
- Then, find the frequency of the above frequencies.
- Finally find which frequency appeared the maximum. If there is a conflict in the frequency of two 2 numbers, then choose the smallest one.

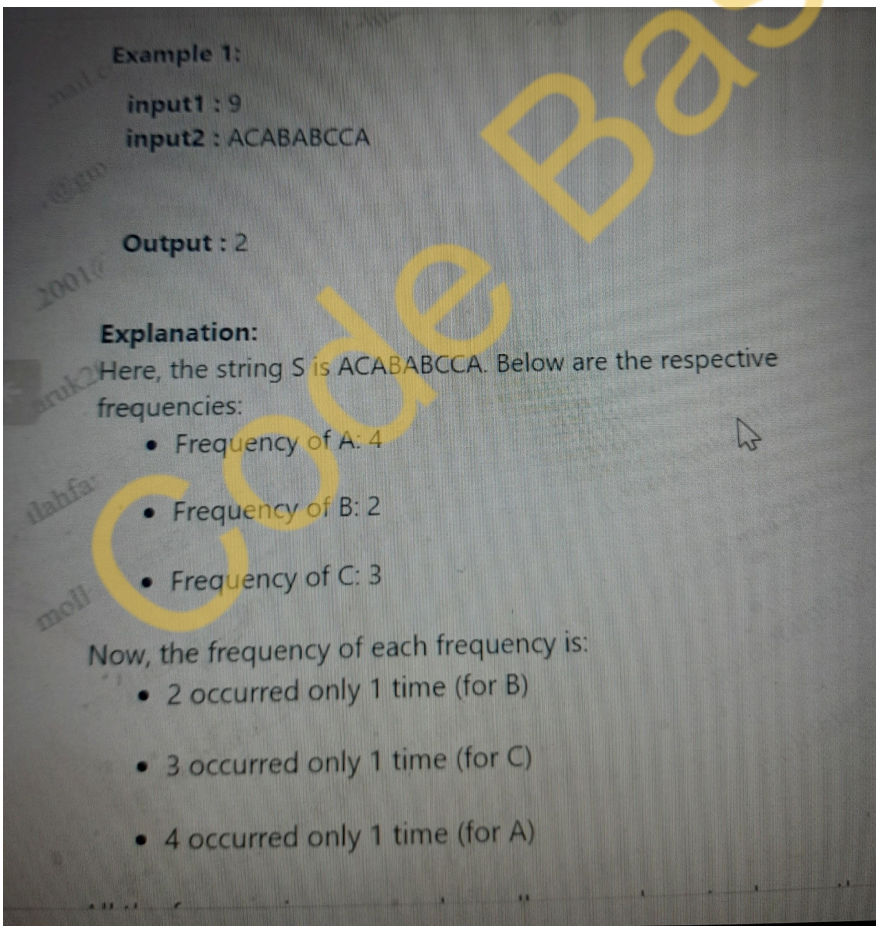
You task is to find and return an integer value representing the frequency which occurred for maximum alphabets.

Input Specification:

input1 : An integer value N, representing the length of string.
input2 : A string value S, representing the alphabets with different frequencies.

Output Specification:

Return an integer value representing the frequency which occurred for maximum alphabets.



Example 1:

input1 : 9
input2 : ACABABCCA

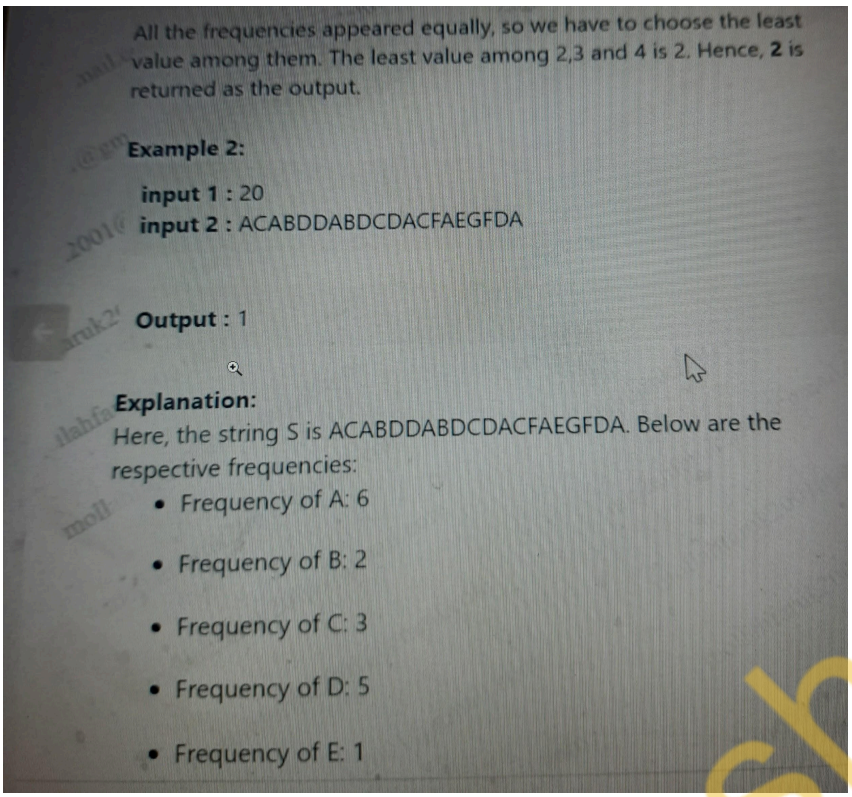
Output : 2

Explanation:
Here, the string S is ACABABCCA. Below are the respective frequencies:

- Frequency of A: 4
- Frequency of B: 2
- Frequency of C: 3

Now, the frequency of each frequency is:

- 2 occurred only 1 time (for B)
- 3 occurred only 1 time (for C)
- 4 occurred only 1 time (for A)


All the frequencies appeared equally, so we have to choose the least value among them. The least value among 2,3 and 4 is 2. Hence, **2** is returned as the output.

Example 2:

input 1 : 20
input 2 : ACABDDABDCDACFAEGFDA

Output : 1

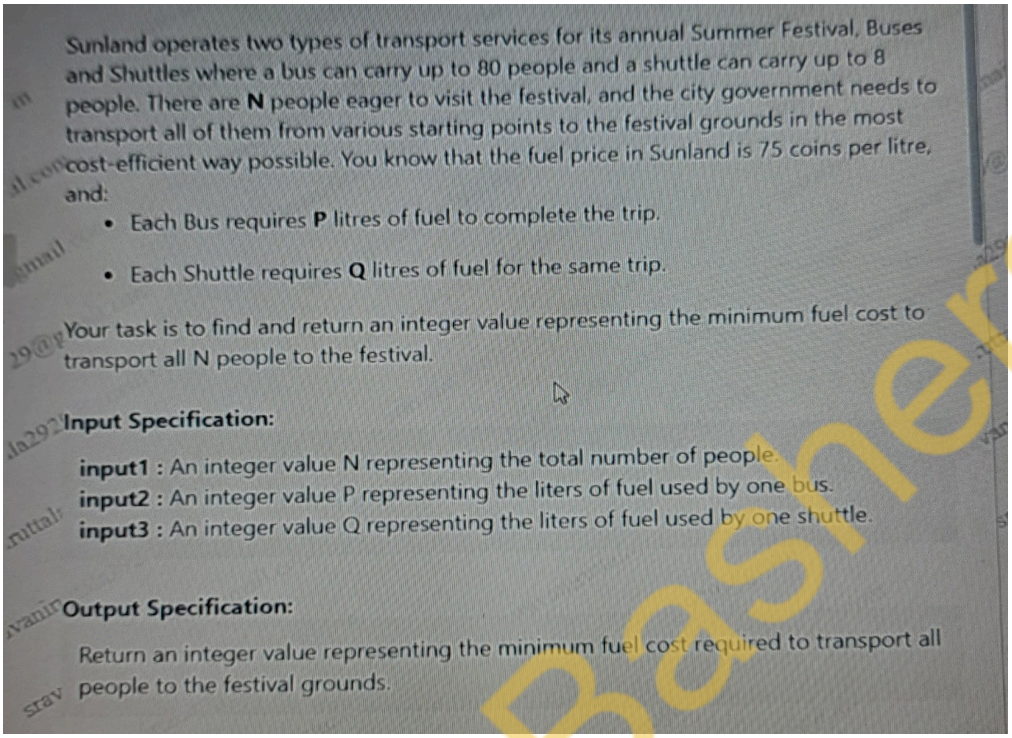
Explanation:
Here, the string S is ACABDDABDCDACFAEGFDA. Below are the respective frequencies:

- Frequency of A: 6
- Frequency of B: 2
- Frequency of C: 3
- Frequency of D: 5
- Frequency of E: 1

```
import java.util.*;

public class Main {
    public static int findMaxFreqFrequency(int N, String S) {
        Map<Character, Integer> freq = new HashMap<>();
        for (char c : S.toCharArray()) {
            freq.put(c, freq.getOrDefault(c, 0) + 1);
        }
        Map<Integer, Integer> freqOfFreq = new HashMap<>();
        for (int f : freq.values()) {
            freqOfFreq.put(f, freqOfFreq.getOrDefault(f, 0) + 1);
        }
        int maxCount = -1, resultFreq = Integer.MAX_VALUE;
        for (Map.Entry<Integer, Integer> e : freqOfFreq.entrySet()) {
            int frequency = e.getKey();
            int count = e.getValue();
            if (count > maxCount || (count == maxCount && frequency < resultFreq)) {
                maxCount = count;
                resultFreq = frequency;
            }
        }
        return resultFreq;
    }

    public static void main(String[] args) {
        int N = 20;
        String S = "ACABDDABDCDACFAEGFDA";
```

```
        System.out.println(findMaxFreqFrequency(N, S)); // Output: 1
    }
}
```

Q16.



Sunland operates two types of transport services for its annual Summer Festival, Buses and Shuttles where a bus can carry up to 80 people and a shuttle can carry up to 8 people. There are **N** people eager to visit the festival, and the city government needs to transport all of them from various starting points to the festival grounds in the most cost-efficient way possible. You know that the fuel price in Sunland is 75 coins per litre, and:

- Each Bus requires **P** litres of fuel to complete the trip.
- Each Shuttle requires **Q** litres of fuel for the same trip.

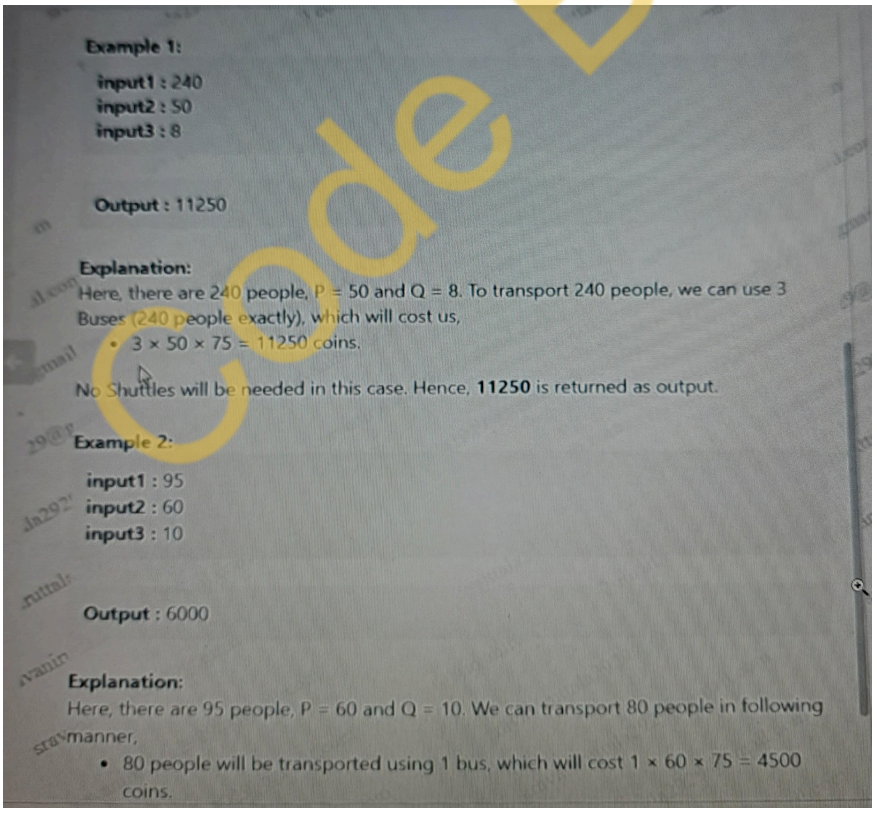
Your task is to find and return an integer value representing the minimum fuel cost to transport all N people to the festival.

Input Specification:

input1 : An integer value N representing the total number of people.
input2 : An integer value P representing the liters of fuel used by one bus.
input3 : An integer value Q representing the liters of fuel used by one shuttle.

Output Specification:

Return an integer value representing the minimum fuel cost required to transport all people to the festival grounds.



Example 1:

input1 : 240
input2 : 50
input3 : 8

Output : 11250

Explanation:
Here, there are 240 people, P = 50 and Q = 8. To transport 240 people, we can use 3 Buses (240 people exactly), which will cost us,

- 3 × 50 × 75 = 11250 coins.

No Shuttles will be needed in this case. Hence, **11250** is returned as output.

Example 2:

input1 : 95
input2 : 60
input3 : 10

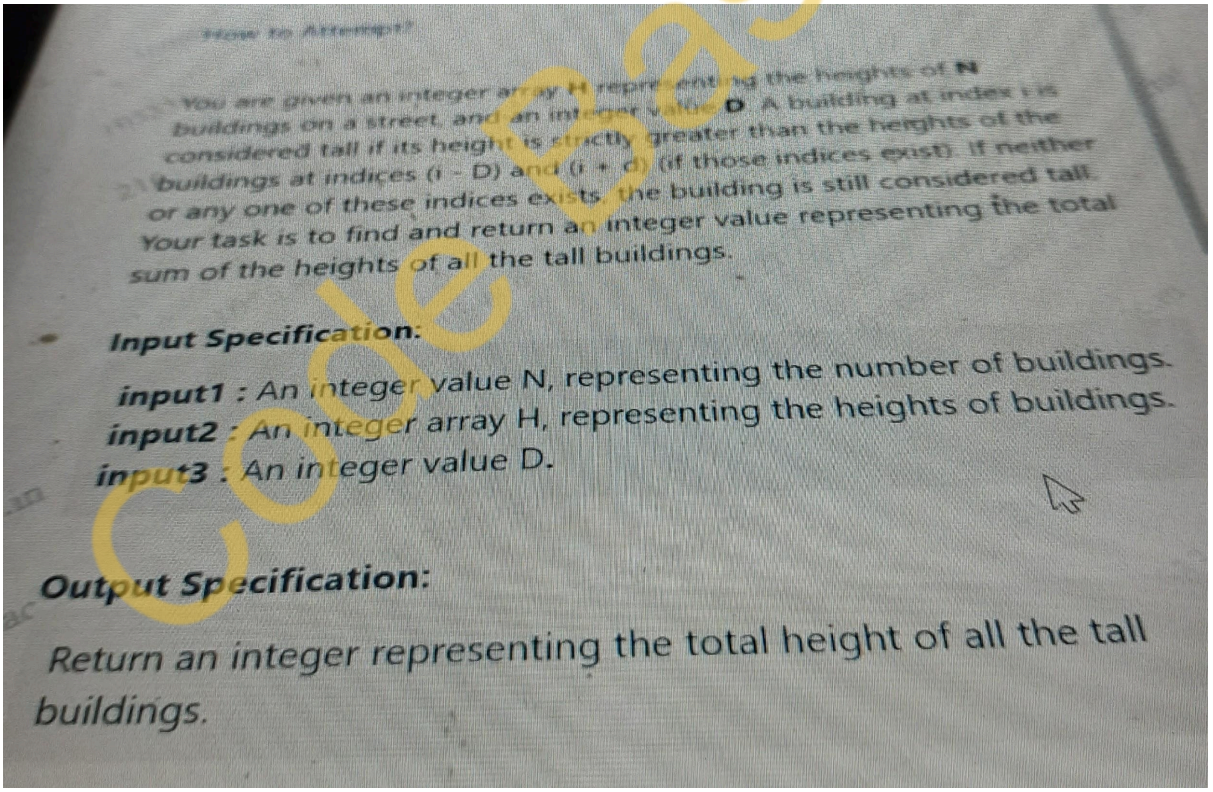
Output : 6000

Explanation:
Here, there are 95 people, P = 60 and Q = 10. We can transport 80 people in following manner,

- 80 people will be transported using 1 bus, which will cost 1 × 60 × 75 = 4500 coins.

```
public class Main {
    public static int minFuelCost(int N, int P, int Q) {
        int busCap = 80, shuttleCap = 8, fuelPrice = 75;
        int minCost = Integer.MAX_VALUE;
        for (int buses = 0; buses <= (N + busCap - 1) / busCap; buses++) {
            int remaining = N - buses * busCap;
            if (remaining < 0) remaining = 0;
            int shuttles = (remaining + shuttleCap - 1) / shuttleCap;
            int cost = (buses * P + shuttles * Q) * fuelPrice;
            minCost = Math.min(minCost, cost);
        }
        return minCost;
    }

    public static void main(String[] args) {
        System.out.println(minFuelCost(95, 60, 10)); // example
    }
}
```

Q17.



How to Attempt?

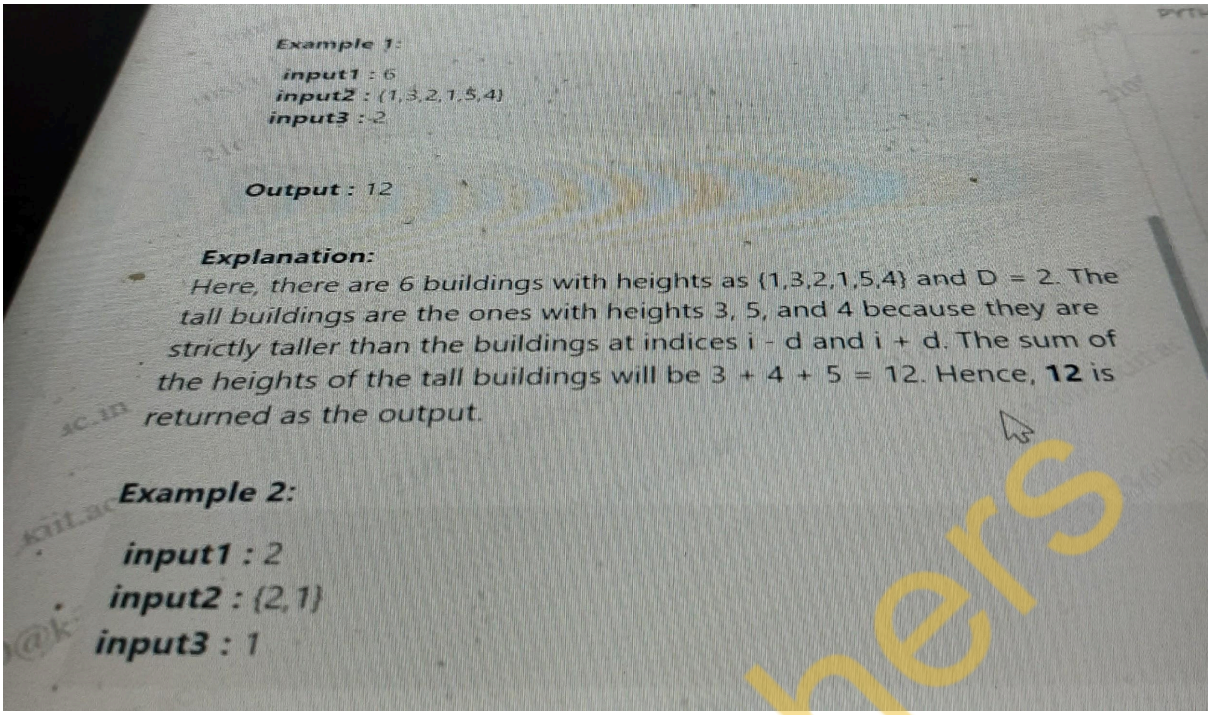
You are given an integer array H representing the heights of N buildings on a street, and an integer value D. A building at index i is considered tall if its height is strictly greater than the herghts of the buildings at indices (i - D) and (i + D) (if those indices exist). If neither or any one of these indices exists, the building is still considered tall. Your task is to find and return an integer value representing the total sum of the heights of all the tall buildings.

Input Specification:

input1 : An integer value N, representing the number of buildings.
input2 : An integer array H, representing the heights of buildings.
input3 : An integer value D.

Output Specification:

Return an integer representing the total height of all the tall buildings.



Example 1:

input1 : 6
input2 : {1,3,2,1,5,4}
input3 : 2

Output : 12

Explanation:

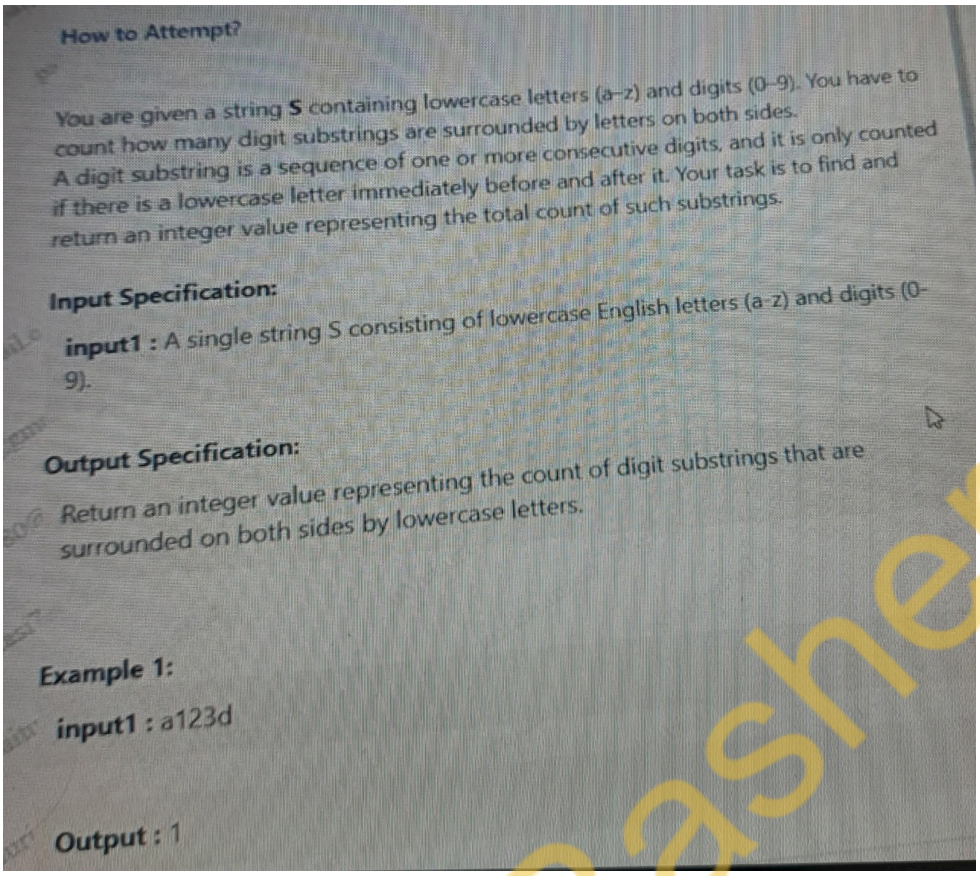
Here, there are 6 buildings with heights as {1,3,2,1,5,4} and D = 2. The tall buildings are the ones with heights 3, 5, and 4 because they are strictly taller than the buildings at indices i - d and i + d. The sum of the heights of the tall buildings will be 3 + 4 + 5 = 12. Hence, **12** is returned as the output.

Example 2:

input1 : 2
input2 : {2,1}
input3 : 1

```
public class Main {
    public static int sumTallBuildings(int N, int[] H, int D) {
        int sum = 0;
        for (int i = 0; i < N; i++) {
            boolean leftOK = (i - D < 0) || (H[i] > H[i - D]);
            boolean rightOK = (i + D >= N) || (H[i] > H[i + D]);
            if (leftOK && rightOK) {
                sum += H[i];
            }
        }
        return sum;
    }

    public static void main(String[] args) {
        int[] H = {1,3,2,1,5,4};
        int D = 2;
        System.out.println(sumTallBuildings(H.length, H, D)); // example
    }
}
```

Q18.



How to Attempt?

You are given a string **S** containing lowercase letters (a–z) and digits (0–9). You have to count how many digit substrings are surrounded by letters on both sides.
A digit substring is a sequence of one or more consecutive digits, and it is only counted if there is a lowercase letter immediately before and after it. Your task is to find and return an integer value representing the total count of such substrings.

Input Specification:

input1 : A single string S consisting of lowercase English letters (a-z) and digits (0-9).

Output Specification:

Return an integer value representing the count of digit substrings that are surrounded on both sides by lowercase letters.

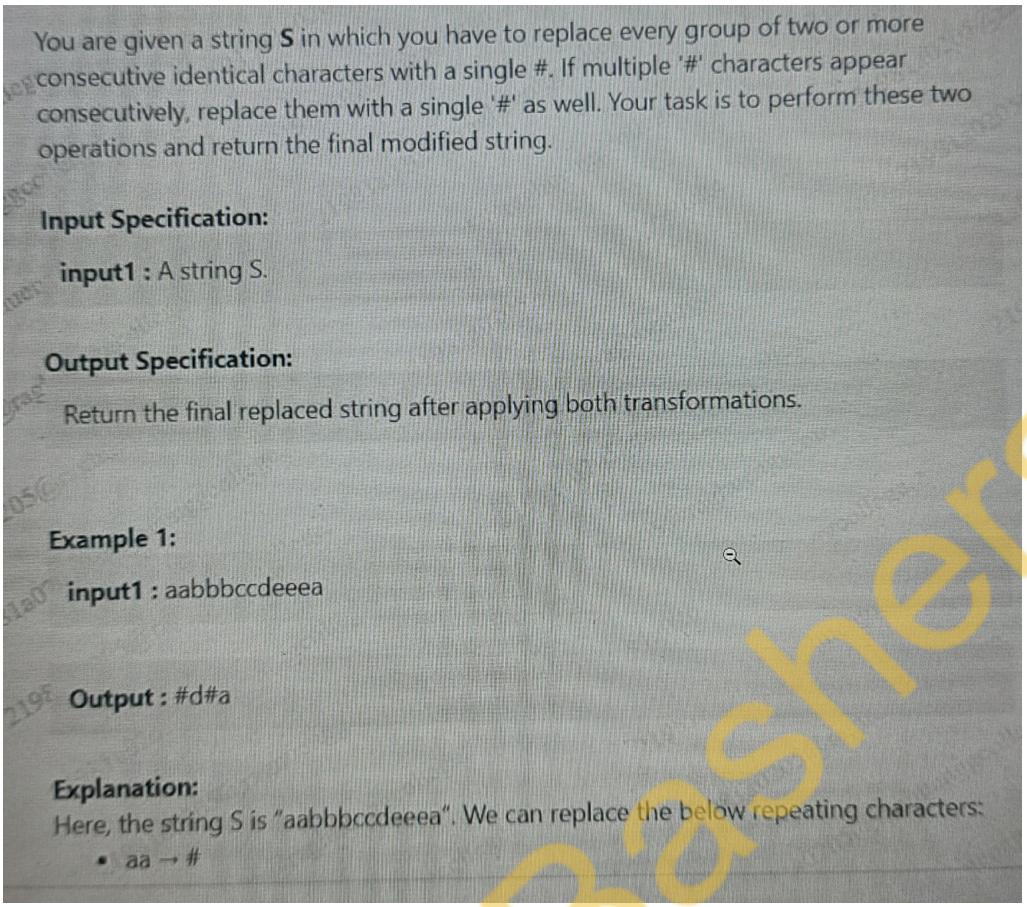
Example 1:

input1 : a123d

Output : 1

```
public class Main {
    public static int countDigitSubstrings(String S) {
        int count = 0, i = 0, n = S.length();
        while (i < n) {
            if (Character.isDigit(S.charAt(i))) {
                int start = i;
                while (i < n && Character.isDigit(S.charAt(i))) i++;
                int end = i - 1;
                if (start > 0 && end < n - 1 && Character.isLetter(S.charAt(start - 1)) &&
Character.isLetter(S.charAt(end + 1))) {
                    count++;
                }
            } else {
                i++;
            }
        }
        return count;
    }

    public static void main(String[] args) {
        String S = "a123d";
        System.out.println(countDigitSubstrings(S));
    }
}
```

Q19.



You are given a string **S** in which you have to replace every group of two or more consecutive identical characters with a single #. If multiple '#' characters appear consecutively, replace them with a single '#' as well. Your task is to perform these two operations and return the final modified string.

Input Specification:

input1 : A string S.

Output Specification:

Return the final replaced string after applying both transformations.

Example 1:

input1 : aabbbccdeeea

Output : #d#a

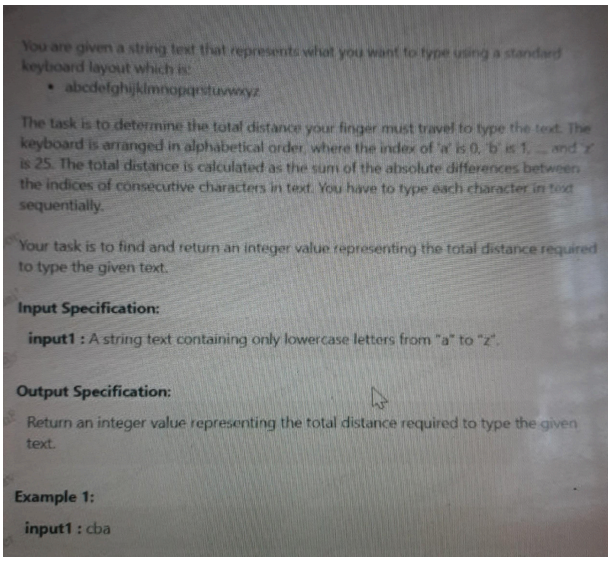
Explanation:

Here, the string S is "aabbbccdeeea". We can replace the below repeating characters:

- aa → #


```
public class Main {
    public static String replaceConsecutive(String S) {
        StringBuilder sb = new StringBuilder();
        int n = S.length();
        int i = 0;
        while (i < n) {
            int j = i;
            while (j < n && S.charAt(j) == S.charAt(i)) j++;
            if (j - i >= 2) {
                if (sb.length() == 0 || sb.charAt(sb.length() - 1) != '#') sb.append('#');
            } else {
                sb.append(S.charAt(i));
            }
            i = j;
        }
        return sb.toString();
    }

    public static void main(String[] args) {
        String S = "aabbbccdeeea";
        System.out.println(replaceConsecutive(S));
    }
}
```

Q20.



You are given a string text that represents what you want to type using a standard keyboard layout which is:

- abcdefghijklmnopqrstuvwxyz

The task is to determine the total distance your finger must travel to type the text. The keyboard is arranged in alphabetical order, where the index of 'a' is 0, 'b' is 1, ..., and 'z' is 25. The total distance is calculated as the sum of the absolute differences between the indices of consecutive characters in text. You have to type each character in text sequentially.

Your task is to find and return an integer value representing the total distance required to type the given text.

Input Specification:

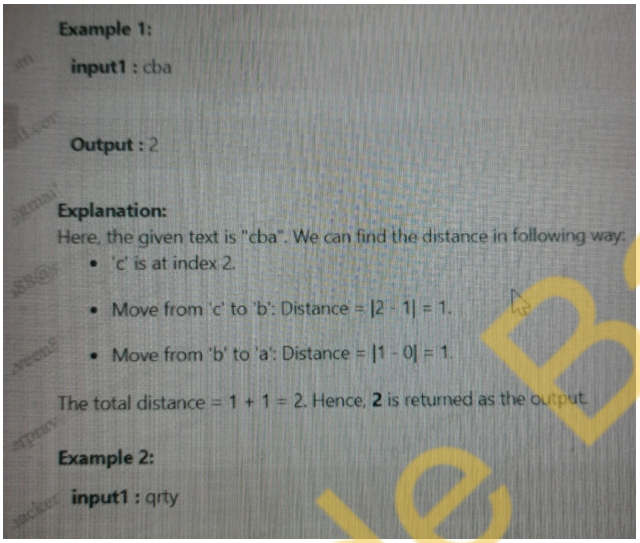
input1 : A string text containing only lowercase letters from "a" to "z".

Output Specification:

Return an integer value representing the total distance required to type the given text.

Example 1:

input1 : cba



Example 1:

input1 : cba

Output : 2

Explanation:

Here, the given text is "cba". We can find the distance in following way:

- 'c' is at index 2.
- Move from 'c' to 'b': Distance = |2 - 1| = 1.
- Move from 'b' to 'a': Distance = |1 - 0| = 1.

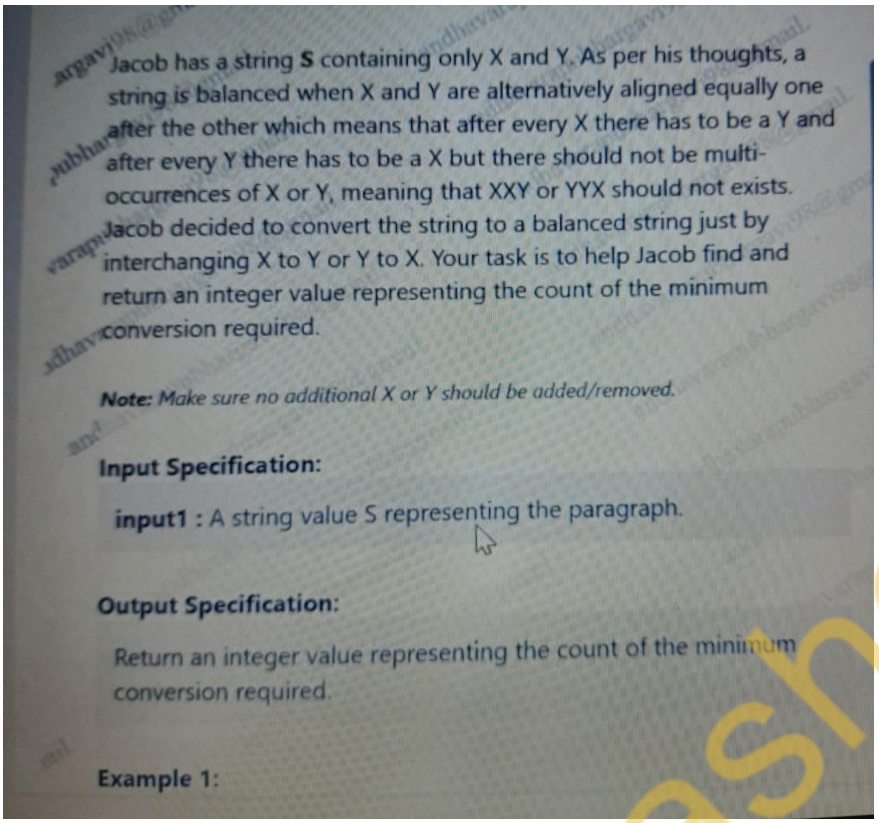
The total distance = 1 + 1 = 2. Hence, **2** is returned as the output.

Example 2:

input1 : qrty

```
public class Main {
    public static int totalTypingDistance(String text) {
        int distance = 0;
        for (int i = 1; i < text.length(); i++) {
            int prevIndex = text.charAt(i - 1) - 'a';
            int currIndex = text.charAt(i) - 'a';
            distance += Math.abs(currIndex - prevIndex);
        }
        return distance;
    }

    public static void main(String[] args) {
        String text = "cba";
        System.out.println(totalTypingDistance(text));
    }
}
```

Q21.



Jacob has a string **S** containing only X and Y. As per his thoughts, a string is balanced when X and Y are alternatively aligned equally one after the other which means that after every X there has to be a Y and after every Y there has to be a X but there should not be multi-occurrences of X or Y, meaning that XXY or YYX should not exists. Jacob decided to convert the string to a balanced string just by interchanging X to Y or Y to X. Your task is to help Jacob find and return an integer value representing the count of the minimum conversion required.

Note: *Make sure no additional X or Y should be added/removed.*

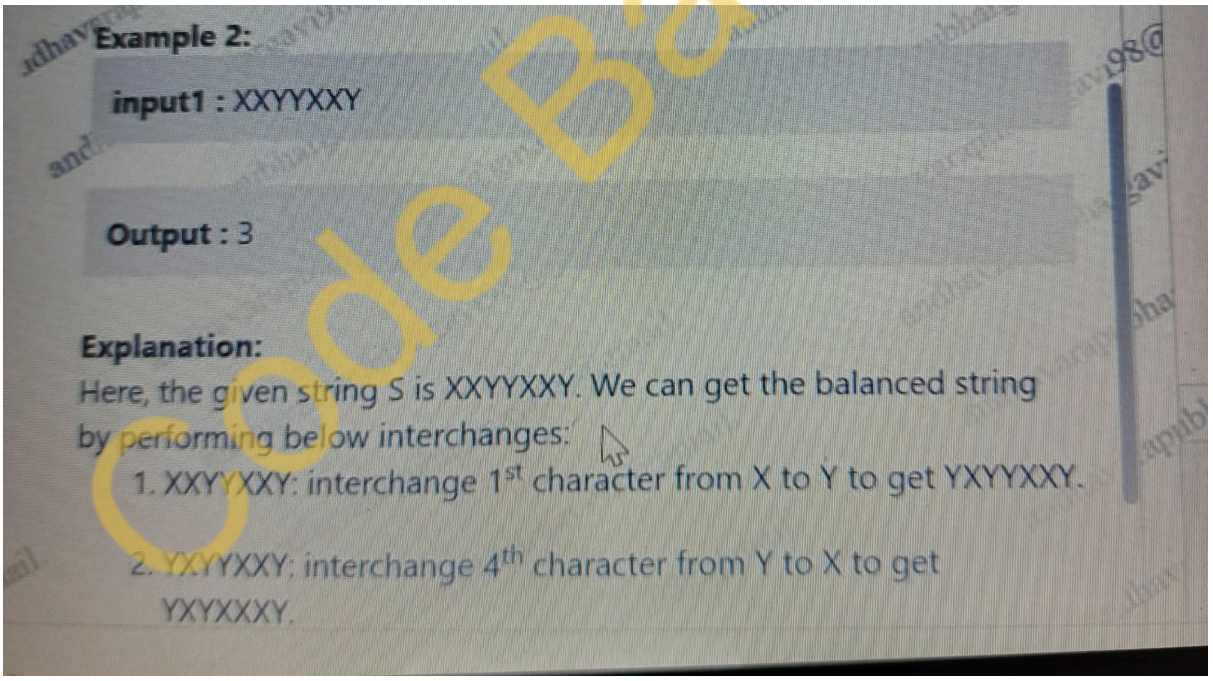
Input Specification:

input1 : A string value S representing the paragraph.

Output Specification:

Return an integer value representing the count of the minimum conversion required.

Example 1:



Example 2:

input1 : XXYYXXY

Output : 3

Explanation:

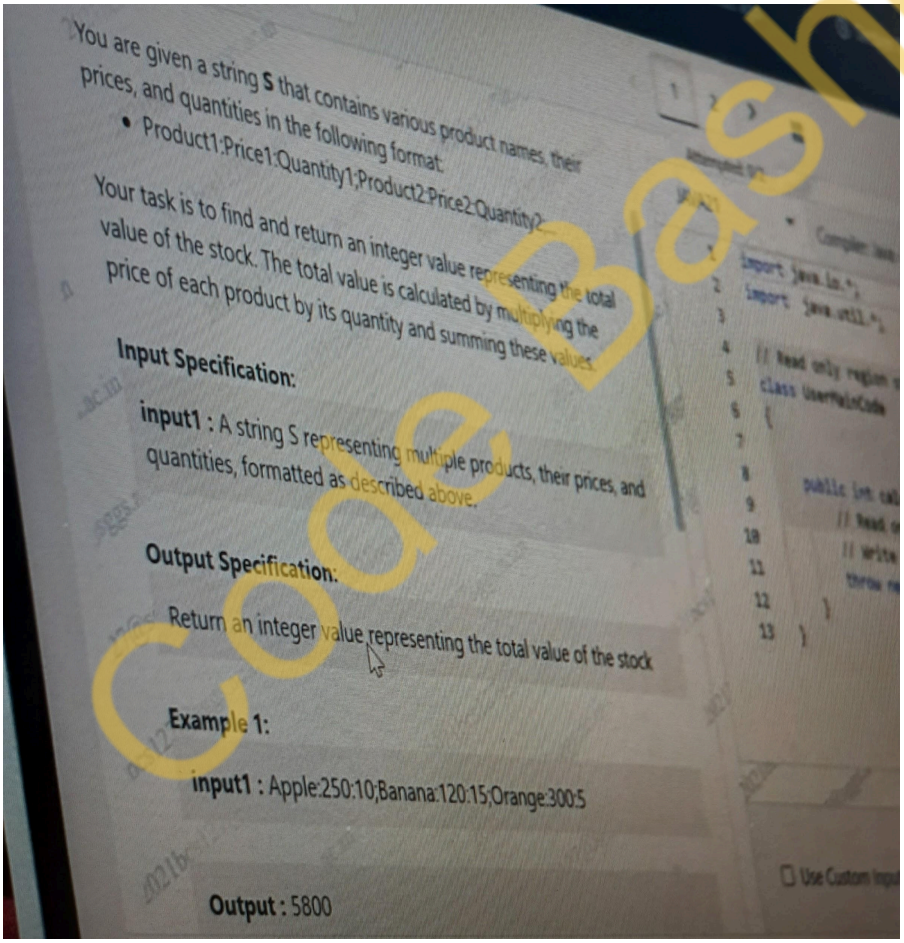
Here, the given string S is XXYYXXY. We can get the balanced string by performing below interchanges:

1. XXYYXXY: interchange 1st character from X to Y to get YXYYXXY.
2. YXYYXXY: interchange 4th character from Y to X to get YXYXXXY.


```
public class Main {
    public static int minConversionsToBalanced(String S) {
        int n = S.length();
        int diff1 = 0; // starting with X
        int diff2 = 0; // starting with Y

        for (int i = 0; i < n; i++) {
            char expected1 = (i % 2 == 0) ? 'X' : 'Y';
```

```
            char expected2 = (i % 2 == 0) ? 'Y' : 'X';

            if (S.charAt(i) != expected1) diff1++;
            if (S.charAt(i) != expected2) diff2++;
        }

        return Math.min(diff1, diff2);
    }

    public static void main(String[] args) {
        String S = "XXYXYY";
        System.out.println(minConversionsToBalanced(S)); // Example output
    }
}
```

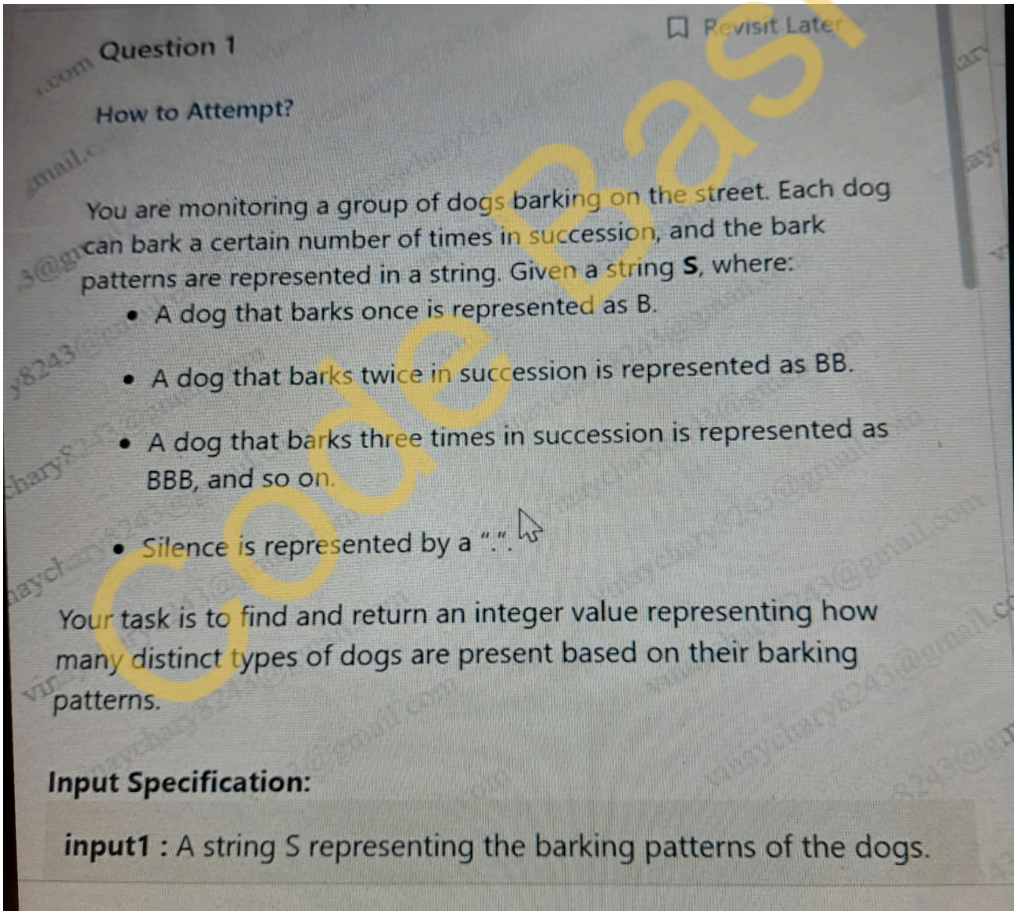
Q22.




```
public class Main {
    public static int totalStockValue(String S) {
        String[] products = S.split(";");
        int totalValue = 0;

        for (String product : products) {
```

```
            String[] parts = product.split(":");
            if (parts.length >= 3) {
                int price = Integer.parseInt(parts[1].trim());
                int quantity = Integer.parseInt(parts[2].trim());
                totalValue += price * quantity;
            }
        }

        return totalValue;
    }

    public static void main(String[] args) {
        String S = "Apple:250:10;Banana:120:15;Orange:300:5";
        System.out.println(totalStockValue(S));
    }
}
```

Q23.



Question 1

Revisit Later

How to Attempt?

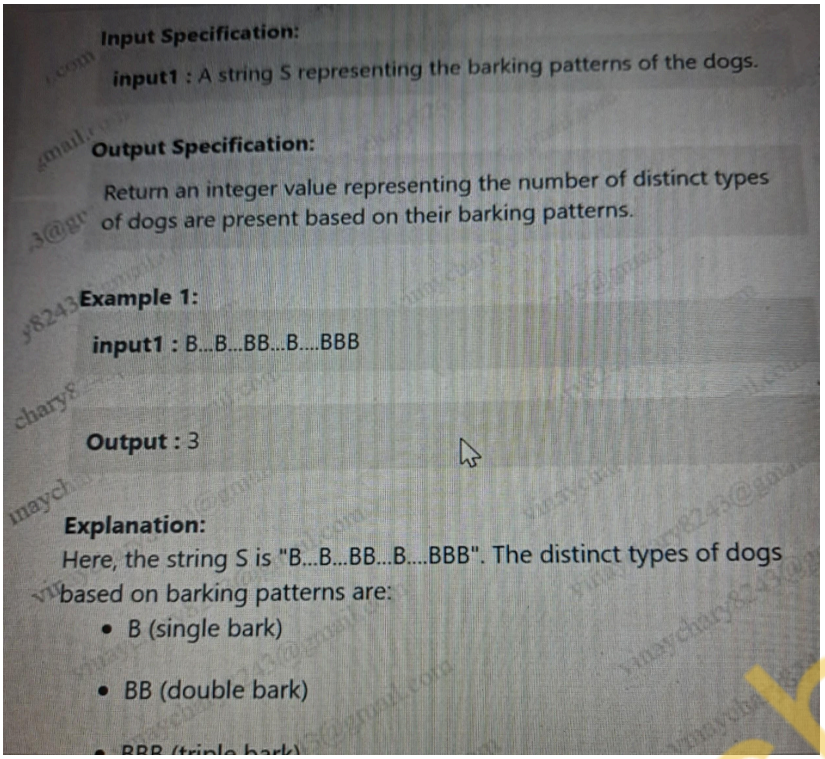
You are monitoring a group of dogs barking on the street. Each dog can bark a certain number of times in succession, and the bark patterns are represented in a string. Given a string **S**, where:

- A dog that barks once is represented as B.
- A dog that barks twice in succession is represented as BB.
- A dog that barks three times in succession is represented as BBB, and so on.
- Silence is represented by a ".".

Your task is to find and return an integer value representing how many distinct types of dogs are present based on their barking patterns.

Input Specification:

input1 : A string S representing the barking patterns of the dogs.



Input Specification:

input1 : A string S representing the barking patterns of the dogs.

Output Specification:

Return an integer value representing the number of distinct types of dogs are present based on their barking patterns.

Example 1:

input1 : B...B...BB...B....BBB

Output : 3

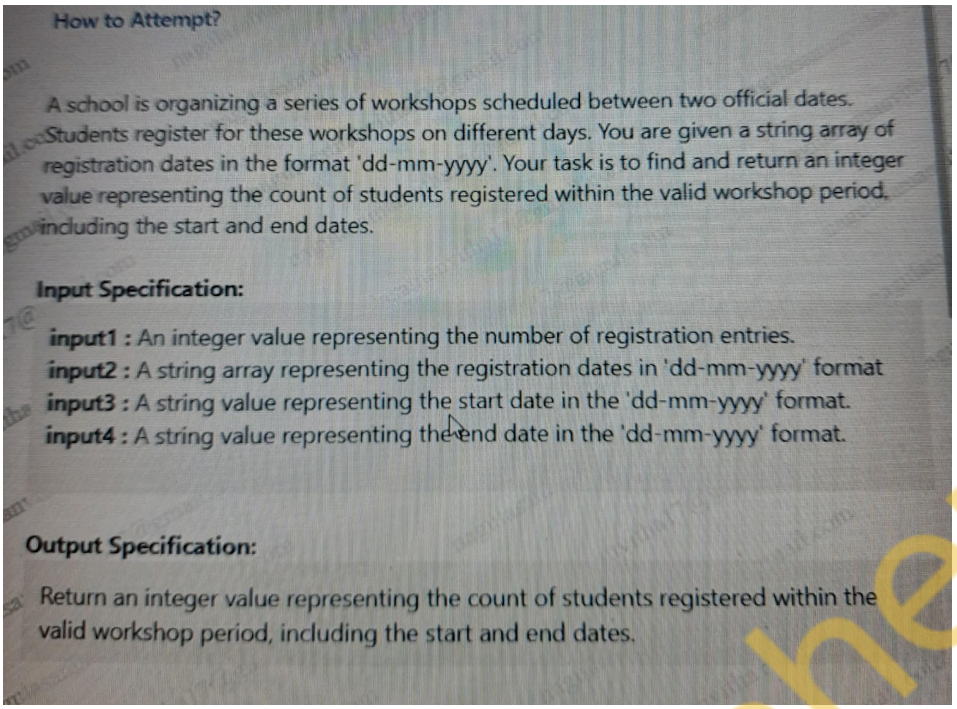
Explanation:

Here, the string S is "B...B...BB...B....BBB". The distinct types of dogs based on barking patterns are:

- B (single bark)
- BB (double bark)
- BBB (triple bark)

```
import java.util.*;

public class Main {
    public static int distinctDogs(String S) {
        String[] parts = S.split("\\.+"); // split by one or more dots
        Set<Integer> barkTypes = new HashSet<>();

        for (String part : parts) {
            if (!part.isEmpty()) {
                barkTypes.add(part.length()); // number of 'B's
            }
        }

        return barkTypes.size();
    }

    public static void main(String[] args) {
        String S = "B...B...BB...B....BBB";
        System.out.println(distinctDogs(S)); // Output: 3
    }
}
```

Q24.



How to Attempt?

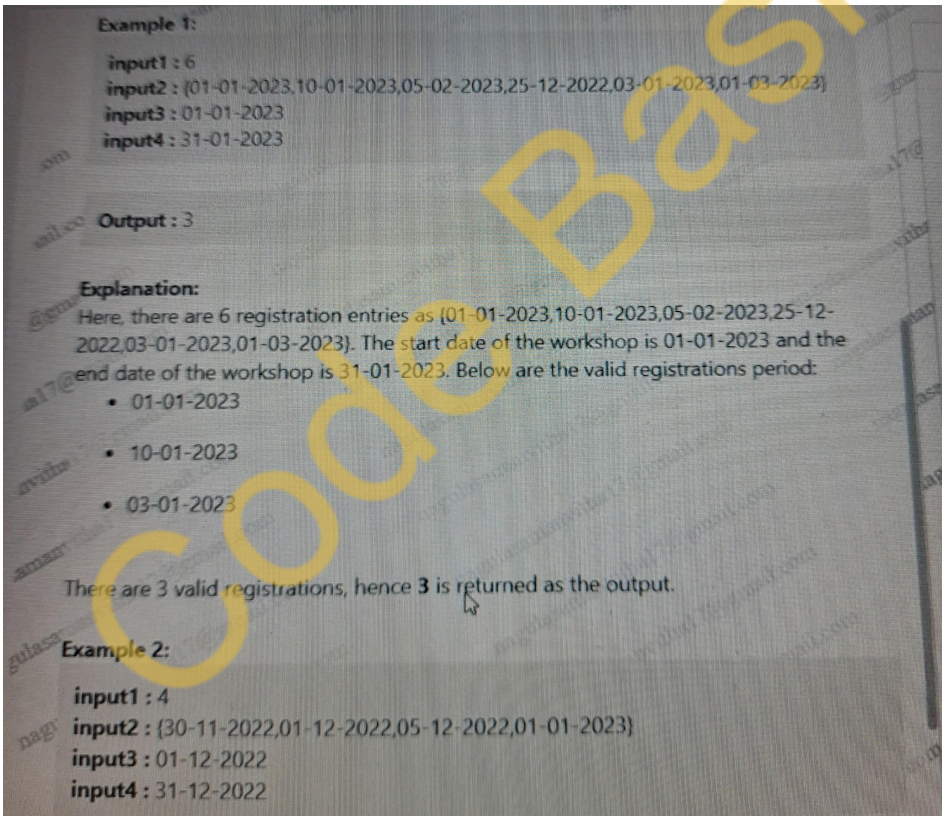
A school is organizing a series of workshops scheduled between two official dates. Students register for these workshops on different days. You are given a string array of registration dates in the format 'dd-mm-yyyy'. Your task is to find and return an integer value representing the count of students registered within the valid workshop period, including the start and end dates.

Input Specification:

input1 : An integer value representing the number of registration entries.
input2 : A string array representing the registration dates in 'dd-mm-yyyy' format
input3 : A string value representing the start date in the 'dd-mm-yyyy' format.
input4 : A string value representing the end date in the 'dd-mm-yyyy' format.

Output Specification:

Return an integer value representing the count of students registered within the valid workshop period, including the start and end dates.



Example 1:

input1 : 6
input2 : {01-01-2023,10-01-2023,05-02-2023,25-12-2022,03-01-2023,01-03-2023}
input3 : 01-01-2023
input4 : 31-01-2023

Output : 3

Explanation:
Here, there are 6 registration entries as {01-01-2023,10-01-2023,05-02-2023,25-12-2022,03-01-2023,01-03-2023}. The start date of the workshop is 01-01-2023 and the end date of the workshop is 31-01-2023. Below are the valid registrations period:

- 01-01-2023
- 10-01-2023
- 03-01-2023

There are 3 valid registrations, hence **3** is returned as the output.

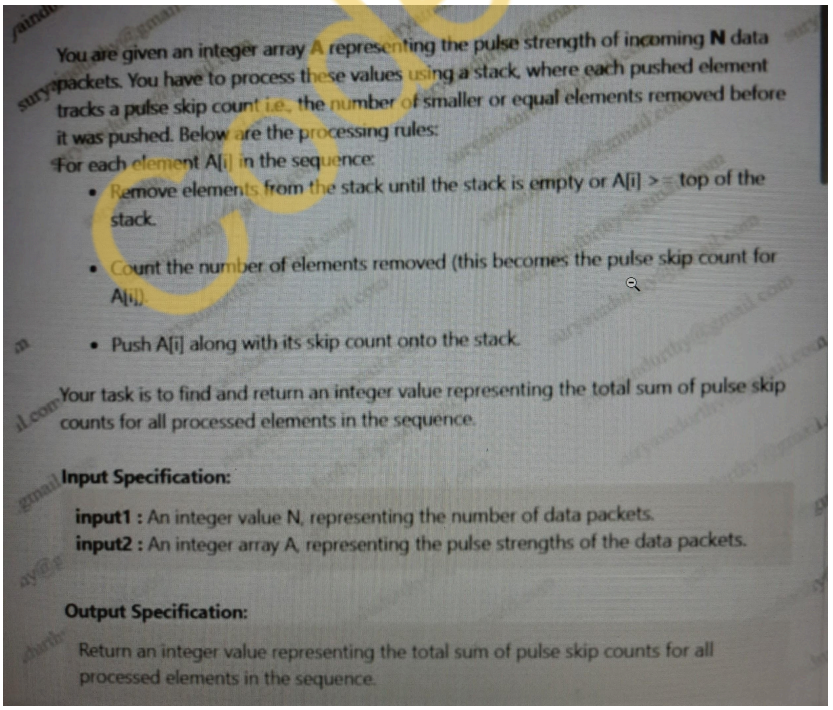
Example 2:

input1 : 4
input2 : {30-11-2022,01-12-2022,05-12-2022,01-01-2023}
input3 : 01-12-2022
input4 : 31-12-2022


```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class Main {
    public static int countValidRegistrations(int n, String[] registrations, String start, String end)
{
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");
```

```
        LocalDate startDate = LocalDate.parse(start, formatter);
        LocalDate endDate = LocalDate.parse(end, formatter);

        int count = 0;
        for (String reg : registrations) {
            LocalDate regDate = LocalDate.parse(reg, formatter);
            if ((regDate.isEqual(startDate) || regDate.isAfter(startDate)) &&
                (regDate.isEqual(endDate) || regDate.isBefore(endDate))) {
                count++;
            }
        }
        return count;
    }

    public static void main(String[] args) {
        int n = 6;
        String[] registrations = {
            "01-01-2023", "10-01-2023", "05-02-2023",
            "25-12-2022", "03-01-2023", "01-03-2023"
        };
        String start = "01-01-2023";
        String end = "31-01-2023";

        System.out.println(countValidRegistrations(n, registrations, start, end)); // Output: 3
    }
}
```

Q25.



You are given an integer array A representing the pulse strength of incoming **N** data packets. You have to process these values using a stack, where each pushed element tracks a pulse skip count i.e., the number of smaller or equal elements removed before it was pushed. Below are the processing rules:

For each element A[i] in the sequence:

- Remove elements from the stack until the stack is empty or A[i] > = top of the stack.
- Count the number of elements removed (this becomes the pulse skip count for A[i]).
- Push A[i] along with its skip count onto the stack.

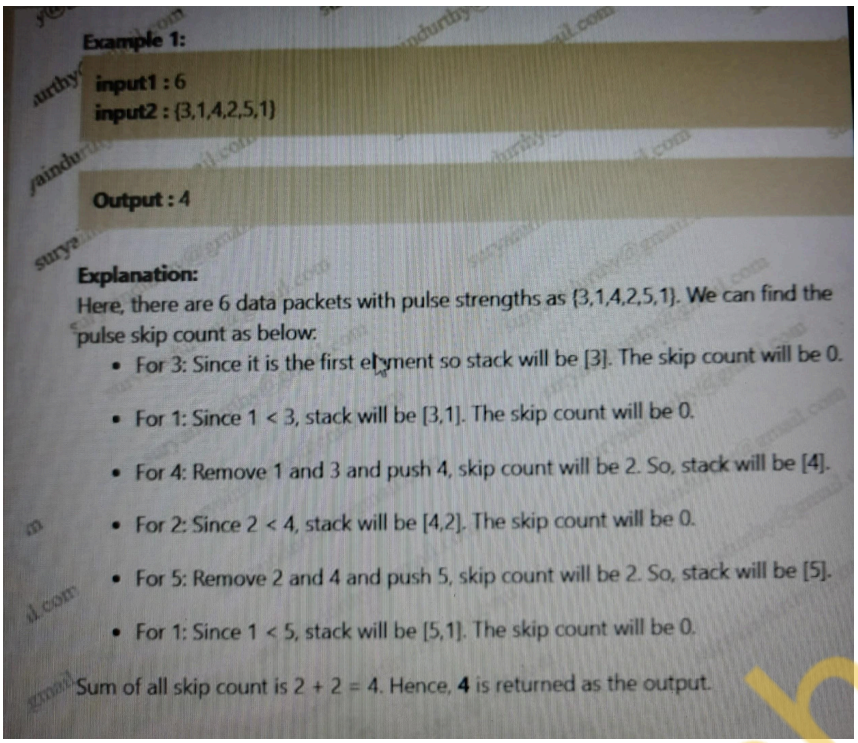
Your task is to find and return an integer value representing the total sum of pulse skip counts for all processed elements in the sequence.

Input Specification:

input1 : An integer value N, representing the number of data packets.
input2 : An integer array A, representing the pulse strengths of the data packets.

Output Specification:

Return an integer value representing the total sum of pulse skip counts for all processed elements in the sequence.



Example 1:

input1 : 6
input2 : {3,1,4,2,5,1}

Output : 4

Explanation:

Here, there are 6 data packets with pulse strengths as {3,1,4,2,5,1}. We can find the pulse skip count as below:

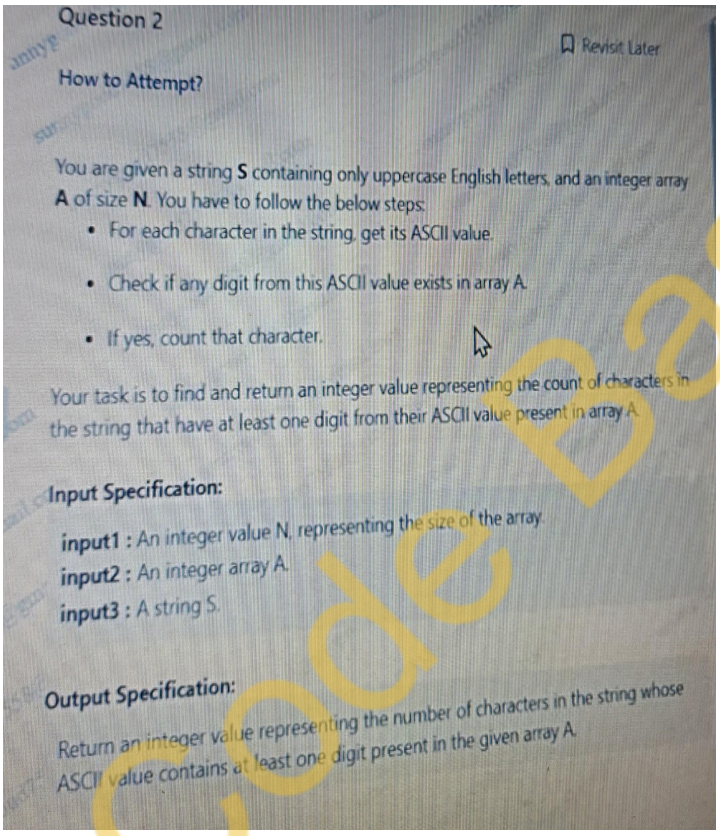
- For 3: Since it is the first element so stack will be [3]. The skip count will be 0.
- For 1: Since 1 < 3, stack will be [3,1]. The skip count will be 0.
- For 4: Remove 1 and 3 and push 4, skip count will be 2. So, stack will be [4].
- For 2: Since 2 < 4, stack will be [4,2]. The skip count will be 0.
- For 5: Remove 2 and 4 and push 5, skip count will be 2. So, stack will be [5].
- For 1: Since 1 < 5, stack will be [5,1]. The skip count will be 0.

Sum of all skip count is 2 + 2 = 4. Hence, **4** is returned as the output.

```
import java.util.Stack;

public class Solution {
    static class Element {
        int value;
        int skipCount;

        Element(int value, int skipCount) {
            this.value = value;
            this.skipCount = skipCount;
        }
    }

    public static int calculateTotalSkipCount(int n, int[] a) {
        Stack<Element> stack = new Stack<>();
        int totalSkipCount = 0;

        for (int i = 0; i < n; i++) {
            int currentSkipCount = 0;

            while (!stack.isEmpty() && a[i] > stack.peek().value) {
                currentSkipCount++;
                stack.pop();
            }

            stack.push(new Element(a[i], currentSkipCount));
            totalSkipCount += currentSkipCount;
        }
```

```
        return totalSkipCount;
    }

    public static void main(String[] args) {
        int n = 6;
        int[] a = {3, 1, 4, 2, 5, 1};
        System.out.println(calculateTotalSkipCount(n, a));
    }
}
```

Q26.



Question 2

Revisit Later

How to Attempt?

You are given a string **S** containing only uppercase English letters, and an integer array **A** of size **N**. You have to follow the below steps:

- For each character in the string, get its ASCII value.
- Check if any digit from this ASCII value exists in array A.
- If yes, count that character.

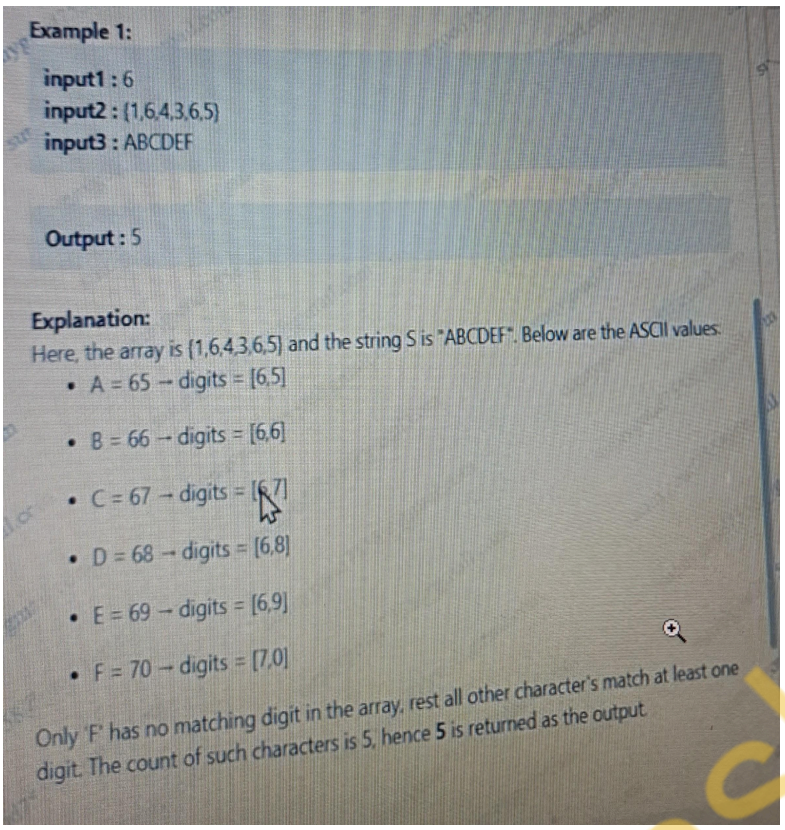
Your task is to find and return an integer value representing the count of characters in the string that have at least one digit from their ASCII value present in array A.

Input Specification:

input1 : An integer value N, representing the size of the array.
input2 : An integer array A.
input3 : A string S.

Output Specification:

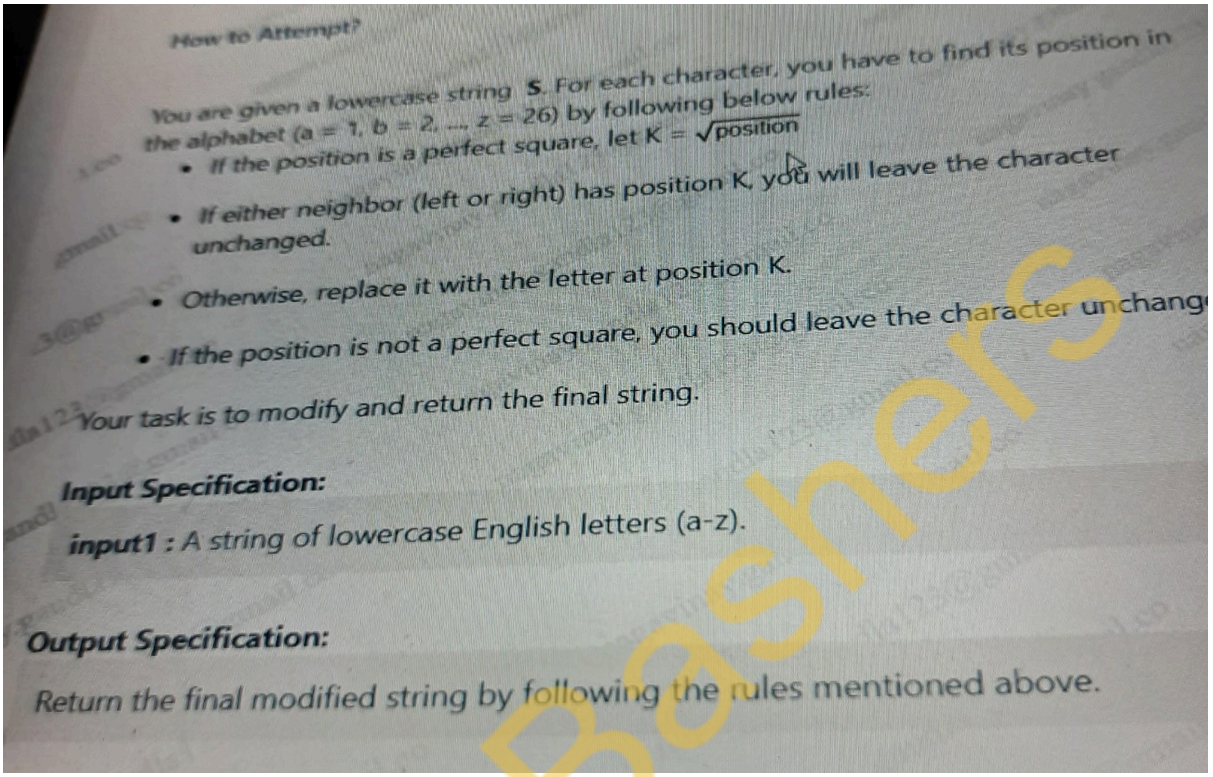
Return an integer value representing the number of characters in the string whose ASCII value contains at least one digit present in the given array A.




```
import java.util.*;

public class Main {
    public static int countChars(int N, int[] A, String S) {
        Set<Integer> allowed = new HashSet<>();
        for (int num : A) allowed.add(num);

        int count = 0;
        for (char ch : S.toCharArray()) {
            int asciiVal = (int) ch;
            String asciiStr = String.valueOf(asciiVal);
            for (char digitChar : asciiStr.toCharArray()) {
                int digit = digitChar - '0';
                if (allowed.contains(digit)) {
                    count++;
                    break; // no need to check other digits
                }
            }
        }
        return count;
    }

    public static void main(String[] args) {
        int[] A = {1,6,4,3,6,5};
        String S = "ABCDEF";
        System.out.println(countChars(3, A, S));
```

```
    }
}
```

Q27.



How to Attempt?

You are given a lowercase string **S**. For each character, you have to find its position in the alphabet (a = 1, b = 2, ..., z = 26) by following below rules:

- If the position is a perfect square, let K = $\sqrt{\text{position}}$
- If either neighbor (left or right) has position K, you will leave the character unchanged.
- Otherwise, replace it with the letter at position K.
- If the position is not a perfect square, you should leave the character unchang

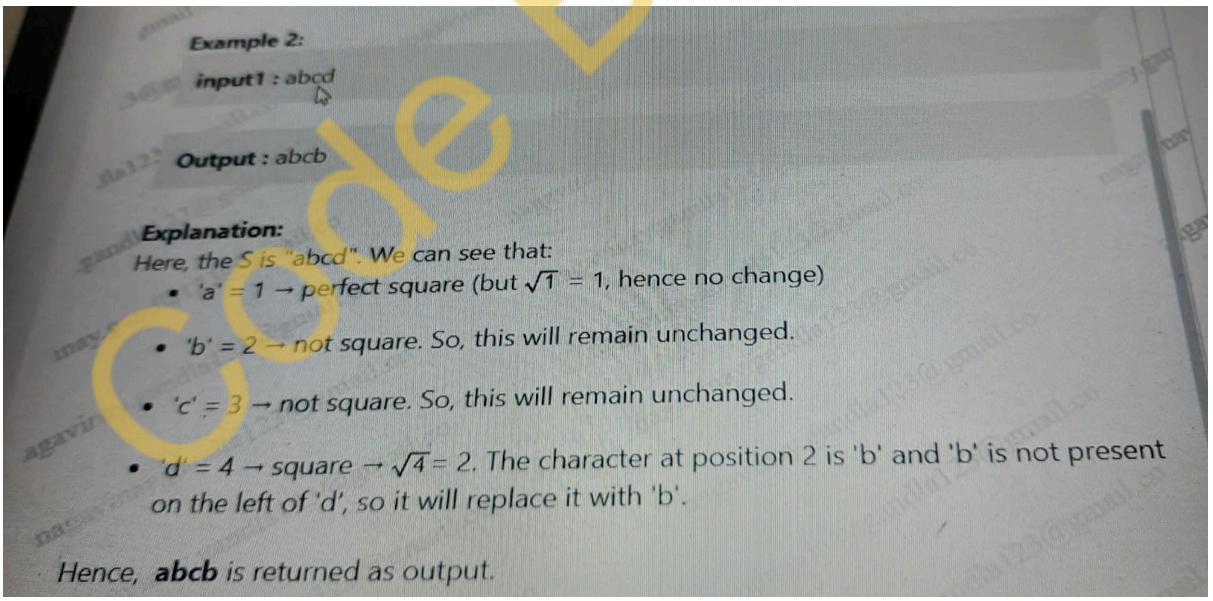
Your task is to modify and return the final string.

Input Specification:

input1 : A string of lowercase English letters (a-z).

Output Specification:

Return the final modified string by following the rules mentioned above.



Example 2:

input1 : abcd

Output : abcb

Explanation:

Here, the S is "abcd". We can see that:

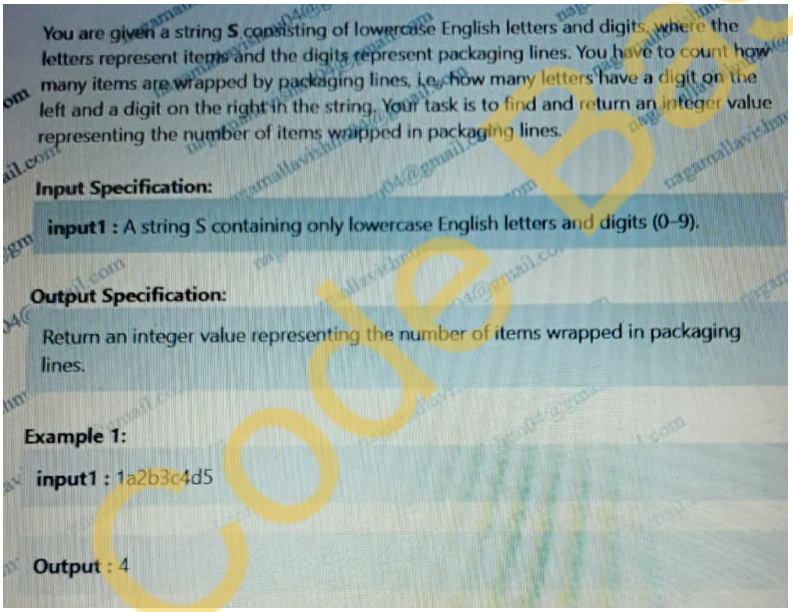
- 'a' = 1 → perfect square (but $\sqrt{1}$ = 1, hence no change)
- 'b' = 2 → not square. So, this will remain unchanged.
- 'c' = 3 → not square. So, this will remain unchanged.
- 'd' = 4 → square → $\sqrt{4}$ = 2. The character at position 2 is 'b' and 'b' is not present on the left of 'd', so it will replace it with 'b'.

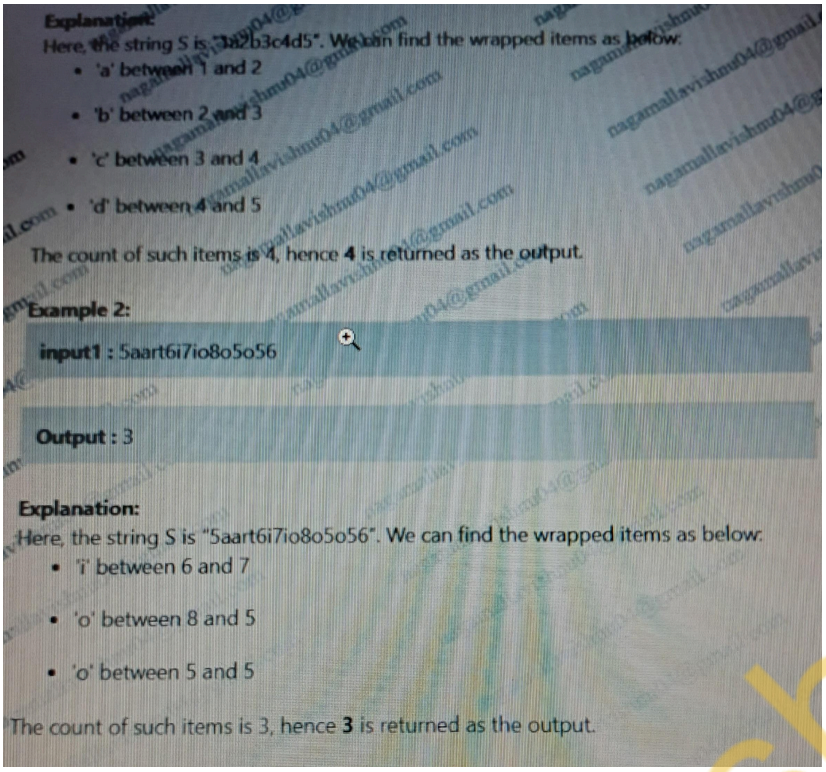
Hence, **abcb** is returned as output.


```
public class Main {
    public static String modifyString(String s) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < s.length(); i++) {
            int p = s.charAt(i) - 'a' + 1;
            if (p == 1 || p == 4 || p == 9 || p == 16 || p == 25) {
                int k = (int) Math.sqrt(p);
```

```
            boolean left = i > 0 && (s.charAt(i - 1) - 'a' + 1) == k;
            boolean right = i < s.length() - 1 && (s.charAt(i + 1) - 'a' + 1) == k;
            if (left || right) {
                result.append(s.charAt(i));
            } else {
                result.append((char) ('a' + k - 1));
            }
        } else {
            result.append(s.charAt(i));
        }
    }
    return result.toString();
  }

  public static void main(String[] args) {
    System.out.println(modifyString("abcd"));
  }
}
```

Q28.





Explanation:

Here, the string S is "1a2b3c4d5". We can find the wrapped items as below:

- 'a' between 1 and 2
- 'b' between 2 and 3
- 'c' between 3 and 4
- 'd' between 4 and 5

The count of such items is 4, hence **4** is returned as the output.

Example 2:

input1 : 5aart6i7io8o5o56

Output : 3

Explanation:

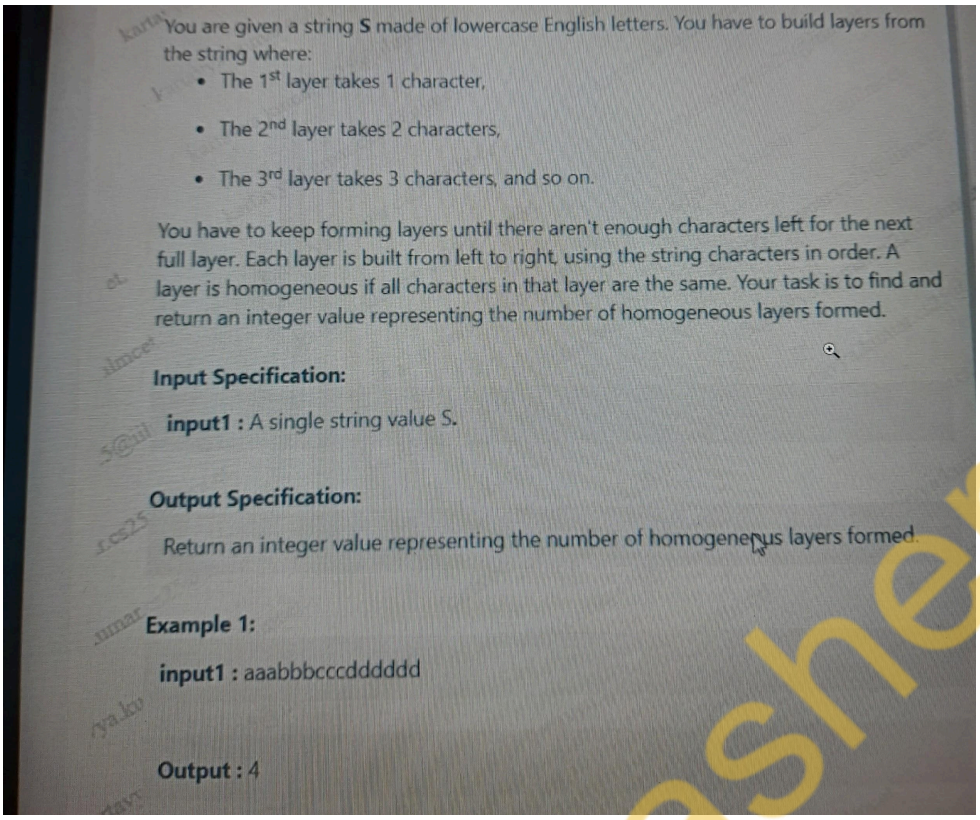
Here, the string S is "5aart6i7io8o5o56". We can find the wrapped items as below:

- 'i' between 6 and 7
- 'o' between 8 and 5
- 'o' between 5 and 5

The count of such items is 3, hence **3** is returned as the output.

```
public class Main {
    public static int countWrappedItems(String s) {
        int count = 0;
        for (int i = 1; i < s.length() - 1; i++) {
            if (Character.isDigit(s.charAt(i - 1)) &&
                Character.isLowerCase(s.charAt(i)) &&
                Character.isDigit(s.charAt(i + 1))) {
                count++;
            }
        }
        return count;
    }

    public static void main(String[] args) {
        String s = "5aart6i7io8o5o56";
        System.out.println(countWrappedItems(s));
    }
}
```

Q29.



You are given a string **S** made of lowercase English letters. You have to build layers from the string where:

- The 1st layer takes 1 character,
- The 2nd layer takes 2 characters,
- The 3rd layer takes 3 characters, and so on.

You have to keep forming layers until there aren't enough characters left for the next full layer. Each layer is built from left to right, using the string characters in order. A layer is homogeneous if all characters in that layer are the same. Your task is to find and return an integer value representing the number of homogeneous layers formed.

Input Specification:

input1 : A single string value S.

Output Specification:

Return an integer value representing the number of homogeneous layers formed.

Example 1:

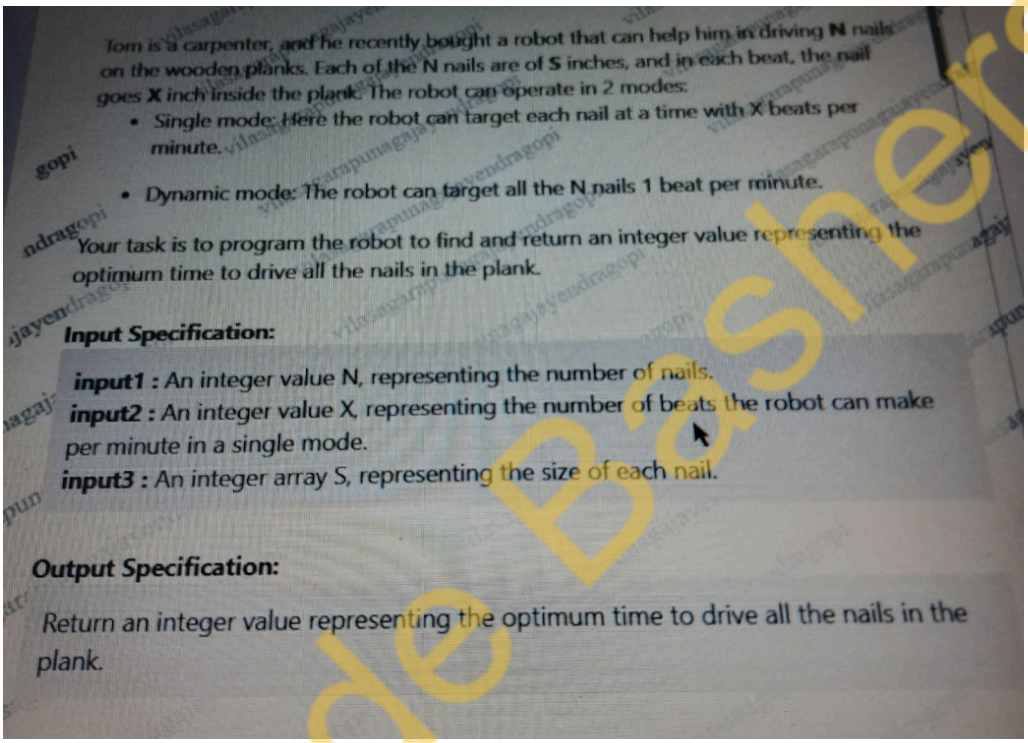
input1 : aaabbbcccdddddd

Output : 4

```
public class Solution {
    public static int countHomogeneousLayers(String s) {
        int homogeneousCount = 0;
        int currentPosition = 0;
        int layerNumber = 1;
        int n = s.length();

        while (currentPosition + layerNumber <= n) {
            boolean isHomogeneous = true;
            char firstChar = s.charAt(currentPosition);

            for (int i = 1; i < layerNumber; i++) {
                if (s.charAt(currentPosition + i) != firstChar) {
                    isHomogeneous = false;
                    break;
                }
            }

            if (isHomogeneous) {
                homogeneousCount++;
            }

            currentPosition += layerNumber;
            layerNumber++;
        }
```

```
        return homogeneousCount;
    }

    public static void main(String[] args) {
        String s = "aaabbbcccdddddd";
        System.out.println(countHomogeneousLayers(s));
    }
}
```

Q30.



Tom is a carpenter, and he recently bought a robot that can help him in driving **N** nails on the wooden planks. Each of the N nails are of **S** inches, and in each beat, the nail goes **X** inch inside the plank. The robot can operate in 2 modes:

- Single mode: Here the robot can target each nail at a time with X beats per minute.
- Dynamic mode: The robot can target all the N nails 1 beat per minute.

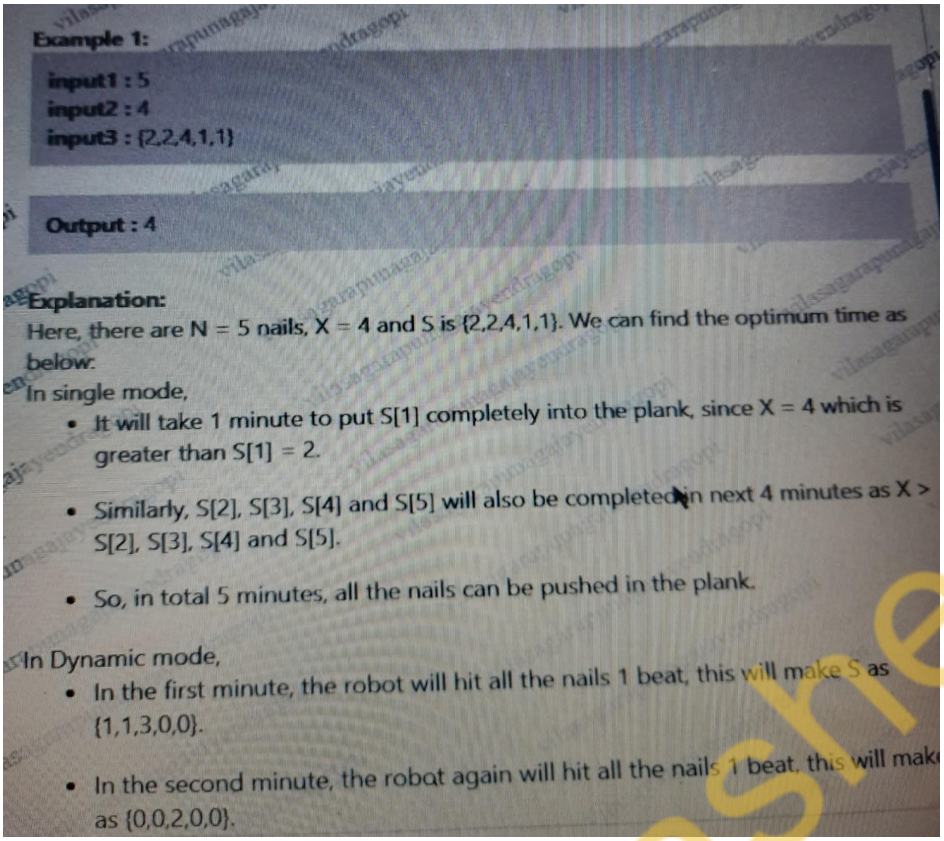
Your task is to program the robot to find and return an integer value representing the optimum time to drive all the nails in the plank.

Input Specification:

input1 : An integer value N, representing the number of nails.
input2 : An integer value X, representing the number of beats the robot can make per minute in a single mode.
input3 : An integer array S, representing the size of each nail.

Output Specification:

Return an integer value representing the optimum time to drive all the nails in the plank.



Example 1:

input1 : 5
input2 : 4
input3 : {2,2,4,1,1}

Output : 4

Explanation:

Here, there are N = 5 nails, X = 4 and S is {2,2,4,1,1}. We can find the optimum time as below:

In single mode,

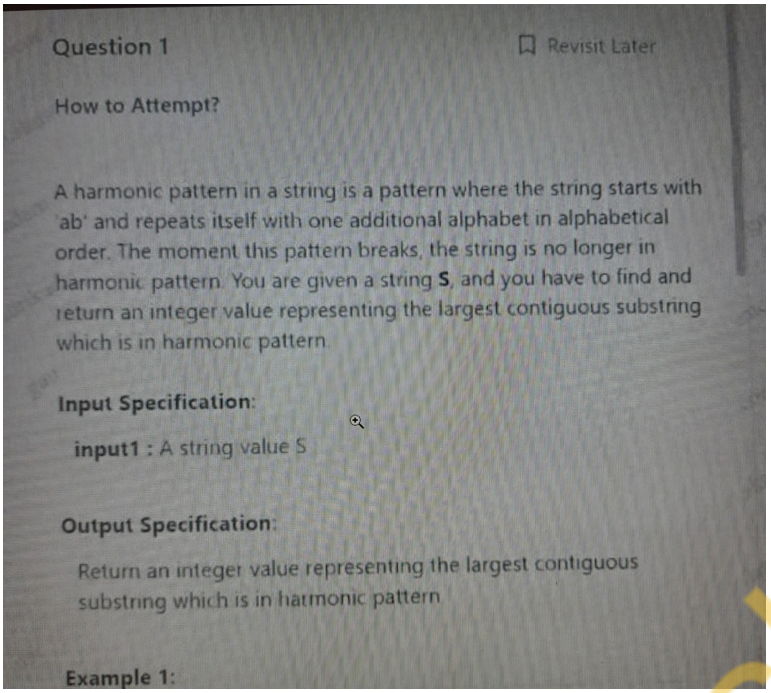
- It will take 1 minute to put S[1] completely into the plank, since X = 4 which is greater than S[1] = 2.
- Similarly, S[2], S[3], S[4] and S[5] will also be completed in next 4 minutes as X > S[2], S[3], S[4] and S[5].
- So, in total 5 minutes, all the nails can be pushed in the plank.

In Dynamic mode,

- In the first minute, the robot will hit all the nails 1 beat, this will make S as {1,1,3,0,0}.
- In the second minute, the robot again will hit all the nails 1 beat, this will mak as {0,0,2,0,0}.

```
public class Solution {
    public static int optimumTime(int N, int X, int[] S) {
        int singleModeTime = 0;
        int maxNailSize = 0;

        for (int i = 0; i < N; i++) {
            singleModeTime += (S[i] + X - 1) / X;
            if (S[i] > maxNailSize) {
                maxNailSize = S[i];
            }
        }

        int dynamicModeTime = maxNailSize;

        return Math.min(singleModeTime, dynamicModeTime);
    }

    public static void main(String[] args) {
        int N = 5;
        int X = 4;
        int[] S = {2, 2, 4, 1, 1};

        System.out.println(optimumTime(N, X, S));
    }
}
```

Q31.



Question 1 Revisit Later

How to Attempt?

A harmonic pattern in a string is a pattern where the string starts with 'ab' and repeats itself with one additional alphabet in alphabetical order. The moment this pattern breaks, the string is no longer in harmonic pattern. You are given a string **S**, and you have to find and return an integer value representing the largest contiguous substring which is in harmonic pattern.

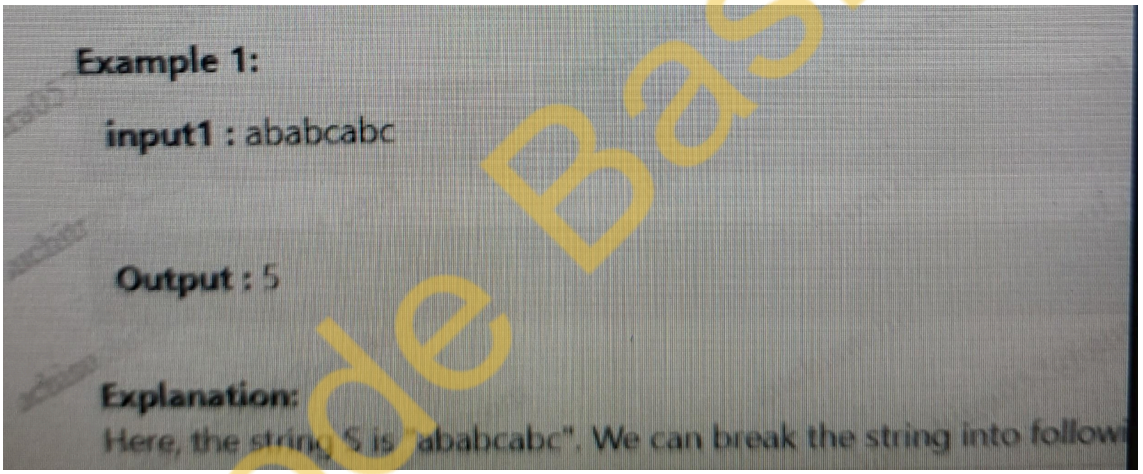
Input Specification:

input1 : A string value S

Output Specification:

Return an integer value representing the largest contiguous substring which is in harmonic pattern

Example 1:



Example 1:

input1 : ababcabc

Output : 5

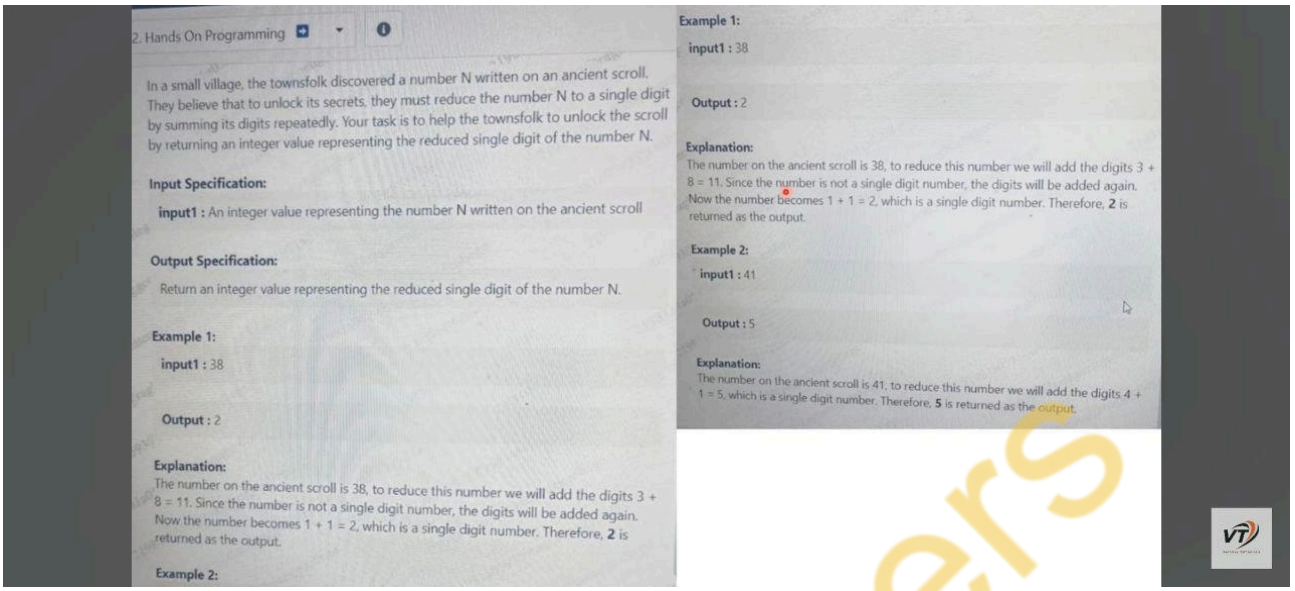
Explanation:

Here, the string S is "ababcabc". We can break the string into follow


```
public class Solution {
    public static int findLargestHarmonicPattern(String s) {
        int maxLength = 0;
        int n = s.length();

        for (int i = 0; i <= n - 2; i++) {
            if (i + 1 < n && s.charAt(i) == 'a' && s.charAt(i + 1) == 'b') {
                int currentLength = getHarmonicLength(s, i);
                maxLength = Math.max(maxLength, currentLength);
            }
        }

        return maxLength;
    }
```

```
    private static int getHarmonicLength(String s, int start) {
        int pos = start;
        int patternLength = 2;
        int totalLength = 0;

        while (pos < s.length()) {
            String expectedPattern = generatePattern(patternLength);

            if (pos + expectedPattern.length() > s.length()) {
                break;
            }

            String actualSubstring = s.substring(pos, pos + expectedPattern.length());

            if (actualSubstring.equals(expectedPattern)) {
                totalLength += expectedPattern.length();
                pos += expectedPattern.length();
                patternLength++;
            } else {
                break;
            }
        }

        return totalLength;
    }

    private static String generatePattern(int length) {
        StringBuilder pattern = new StringBuilder();
        for (int i = 0; i < length; i++) {
            pattern.append((char)('a' + i));
        }
        return pattern.toString();
    }

    public static void main(String[] args) {
        String s = "ababcabc";
        System.out.println(findLargestHarmonicPattern(s));
    }
}
```

Q32.



2. Hands On Programming

In a small village, the townsfolk discovered a number N written on an ancient scroll. They believe that to unlock its secrets, they must reduce the number N to a single digit by summing its digits repeatedly. Your task is to help the townsfolk to unlock the scroll by returning an integer value representing the reduced single digit of the number N.

Input Specification:

input1 : An integer value representing the number N written on the ancient scroll

Output Specification:

Return an integer value representing the reduced single digit of the number N.

Example 1:

input1 : 38

Output : 2

Explanation:
The number on the ancient scroll is 38, to reduce this number we will add the digits 3 + 8 = 11. Since the number is not a single digit number, the digits will be added again. Now the number becomes 1 + 1 = 2, which is a single digit number. Therefore, **2** is returned as the output.

Example 2:

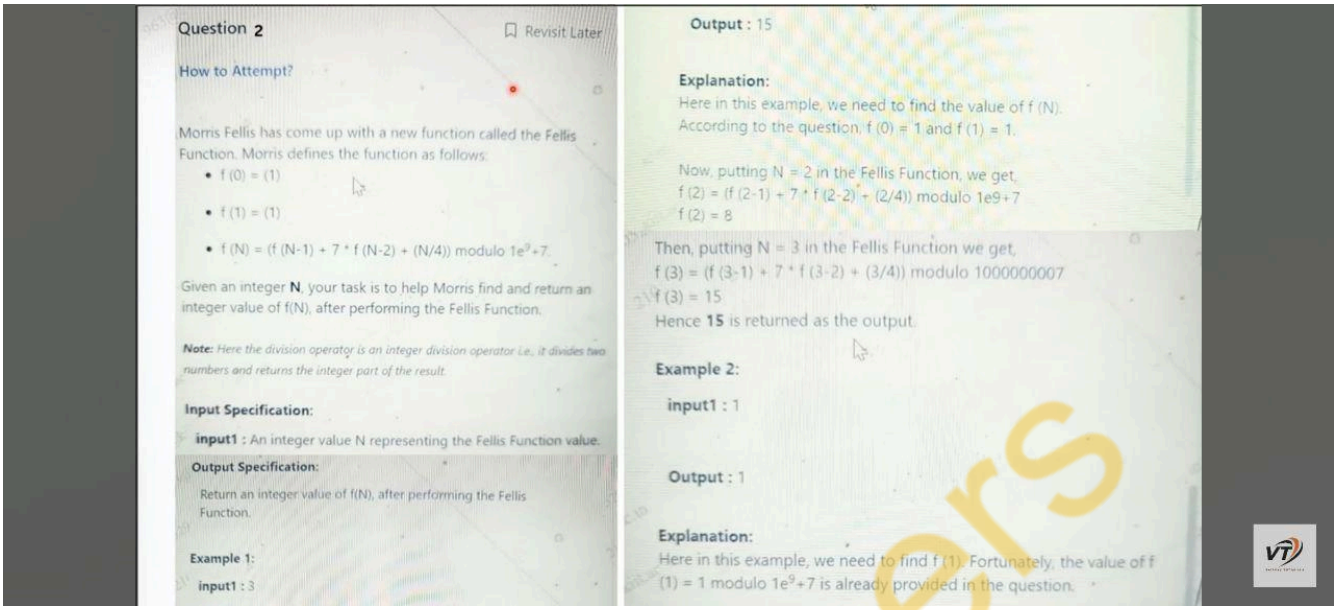
input1 : 41

Output : 5

Explanation:
The number on the ancient scroll is 41, to reduce this number we will add the digits 4 + 1 = 5, which is a single digit number. Therefore, **5** is returned as the output.

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        while (N >= 10) {
            int sum = 0;
            while (N > 0) {
                sum += N % 10;
                N /= 10;
            }
            N = sum;
        }
        System.out.println(N);
    }
}
```

Q33.



Question 2

Revisit Later

How to Attempt?

Morris Fellis has come up with a new function called the Fellis Function. Morris defines the function as follows:

- f (0) = (1)
- f (1) = (1)
- f (N) = (f (N-1) + 7 * f (N-2) + (N/4)) modulo $1e^9+7$

Given an integer **N**, your task is to help Morris find and return an integer value of f(N), after performing the Fellis Function.

Note: *Here the division operator is an integer division operator i.e., it divides two numbers and returns the integer part of the result.*

Input Specification:

input1 : An integer value N representing the Fellis Function value.

Output Specification:

Return an integer value of f(N), after performing the Fellis Function.

Example 1:

input1 : 3

Output : 15

Explanation:

Here in this example, we need to find the value of f (N).
According to the question, f (0) = 1 and f (1) = 1.

Now, putting N = 2 in the Fellis Function, we get,
f (2) = (f (2-1) + 7 * f (2-2) + (2/4)) modulo 1e9+7
f (2) = 8

Then, putting N = 3 in the Fellis Function we get,
f (3) = (f (3-1) + 7 * f (3-2) + (3/4)) modulo 1000000007
f (3) = 15
Hence **15** is returned as the output.

Example 2:

input1 : 1

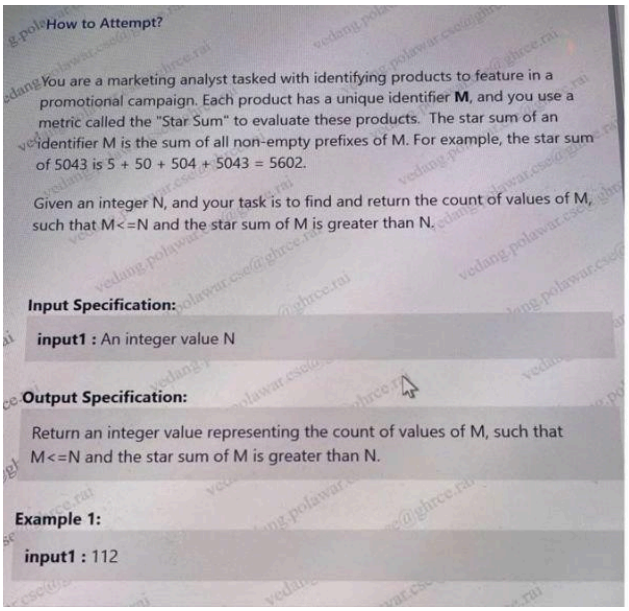
Output : 1

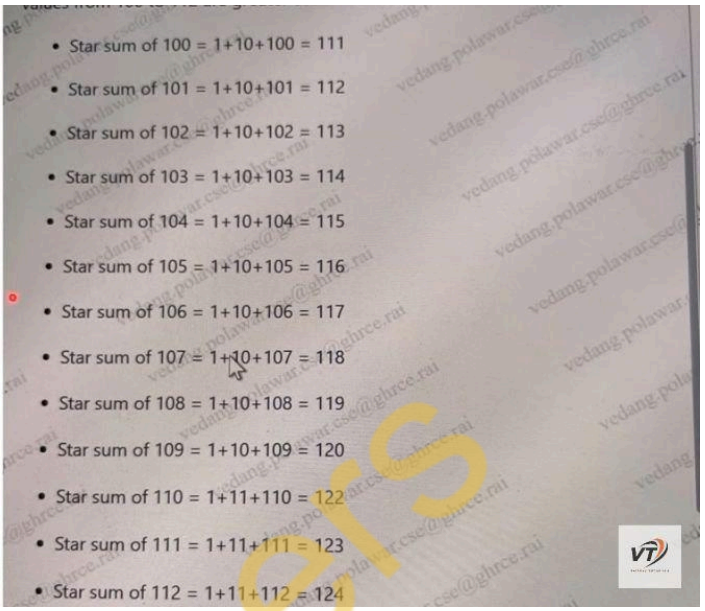
Explanation:

Here in this example, we need to find f (1). Fortunately, the value of f (1) = 1 modulo $1e^9+7$ is already provided in the question.

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        System.out.println(fellis(N));
    }

    static long fellis(int N) {
        long mod = 1000000007;
        if (N == 0 || N == 1) return 1;
        long[] f = new long[N + 1];
        f[0] = 1;
        f[1] = 1;
        for (int i = 2; i <= N; i++) {
            f[i] = (f[i - 1] + 7 * f[i - 2] + (i / 4)) % mod;
        }
        return f[N];
    }
}
```

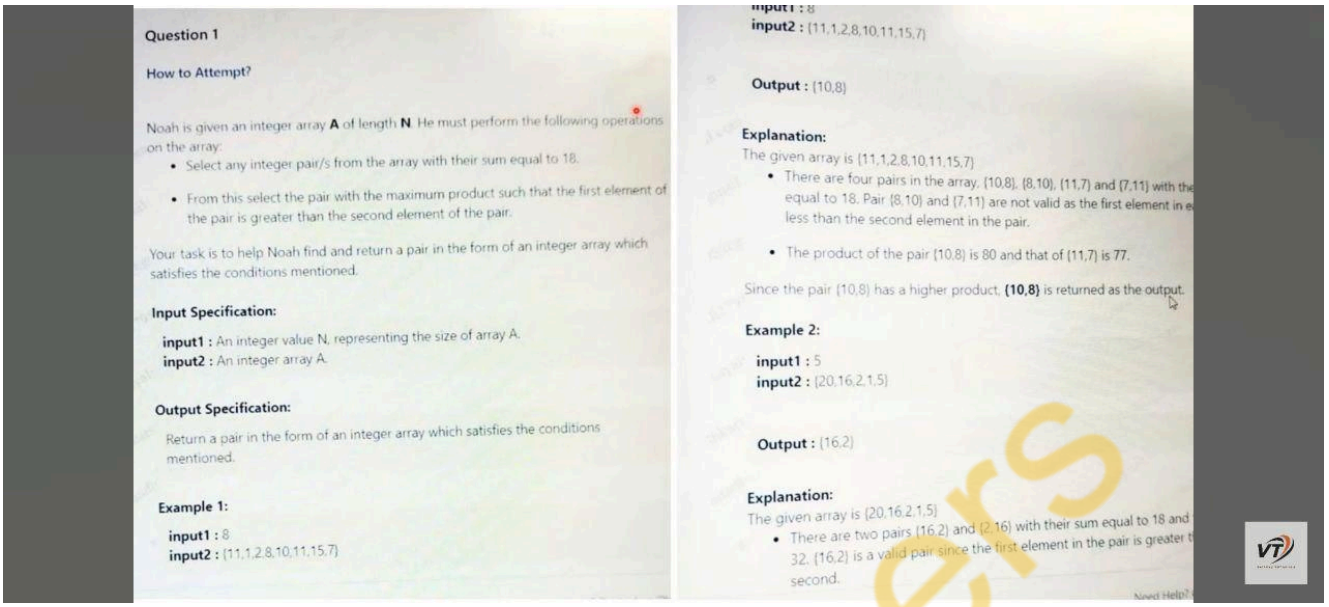
Q34.






```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int count = 0;
        for (int M = 1; M <= N; M++) {
            if (starSum(M) > N) {
                count++;
            }
        }
        System.out.println(count);
    }

    public static int starSum(int num) {
        String s = String.valueOf(num);
        int sum = 0;
        for (int i = 1; i <= s.length(); i++) {
            sum += Integer.parseInt(s.substring(0, i));
        }
        return sum;
    }
}
```

Q35.



Question 1

How to Attempt?

Noah is given an integer array **A** of length **N**. He must perform the following operations on the array:

- Select any integer pair/s from the array with their sum equal to 18.
- From this select the pair with the maximum product such that the first element of the pair is greater than the second element of the pair.

Your task is to help Noah find and return a pair in the form of an integer array which satisfies the conditions mentioned.

Input Specification:

input1 : An integer value N, representing the size of array A.
input2 : An integer array A.

Output Specification:

Return a pair in the form of an integer array which satisfies the conditions mentioned.

Example 1:

input1 : 8
input2 : {11,1,2,8,10,11,15,7}

input1 : 8
input2 : {11,1,2,8,10,11,15,7}

Output : {10,8}

Explanation:
The given array is {11,1,2,8,10,11,15,7}

- There are four pairs in the array, {10,8}, {8,10}, {11,7} and {7,11} with the equal to 18. Pair {8,10} and {7,11} are not valid as the first element in e less than the second element in the pair.
- The product of the pair {10,8} is 80 and that of {11,7} is 77.

Since the pair {10,8} has a higher product, **{10,8}** is returned as the output.

Example 2:

input1 : 5
input2 : {20,16,2,1,5}

Output : {16,2}

Explanation:
The given array is {20,16,2,1,5}

- There are two pairs {16,2} and {2,16} with their sum equal to 18 and 32. {16,2} is a valid pair since the first element in the pair is greater t second.

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int[] A = new int[N];
        for (int i = 0; i < N; i++) {
            A[i] = sc.nextInt();
        }
        int[] result = findPair(A);
        System.out.println(result[0] + " " + result[1]);
    }

    public static int[] findPair(int[] A) {
        int maxProduct = Integer.MIN_VALUE;
        int[] bestPair = new int[2];
        for (int i = 0; i < A.length; i++) {
            for (int j = 0; j < A.length; j++) {
                if (i != j && A[i] + A[j] == 18 && A[i] > A[j]) {
                    int product = A[i] * A[j];
                    if (product > maxProduct) {
                        maxProduct = product;
                        bestPair[0] = A[i];
                        bestPair[1] = A[j];
                    }
                }
            }
        }
        return bestPair;
    }
```

}
Q36.Nth largest element

Sajith loves numbers and coding. His dad gives a task to write a code to find the nth largest number in an array.

Input
The inputs have : First the count of integers and the second n value
The second line of the input has the list of integers that he needs to do the operations upon.

Output
You have to print the integer representing the nth largest out of the numbers given by his father

Constraints
0 < value n <= count of integers <= 10^6
-10^6 <= each integer <= 10^6
0 <= i < count of integers

Example Input
5 3
11 -1 -4 12 -6

Output
-1

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int count = sc.nextInt();
        int n = sc.nextInt();
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        for (int i = 0; i < count; i++) {
            int num = sc.nextInt();
            if (pq.size() < n) pq.add(num);
            else if (num > pq.peek()) {
                pq.poll();
                pq.add(num);
            }
        }
        System.out.println(pq.peek());
    }
}
```

Q37.Playing with geometric progression

Take input from the user for an Geometric Progression, first term and the common ratio will be given, you have to find out the nth term of the G.P

Sample Input and Output

1st line contains the initial term of G.P
2nd line contains the common ratio of G.P
3rd line contains the nth term we have to find
4th line contains the output

Example
2
3
6
486

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        long a = sc.nextLong();
        long r = sc.nextLong();
        long n = sc.nextLong();
        System.out.println(a * (long)Math.pow(r, n - 1));
    }
}
```

Q38.Flipkart Virus

Flipkart has been infected with a virus, the virus works this way: Each user that has been infected with the virus is traced by a special algorithm and a hint that the virus gives. The virus leaves a hint number N. Flipkart is able to identify the user ID of the virus by this N number as the user ID works as a series : Each number in the series is the sum of the last three numbers in the series. The first three numbers in the series are 0, 0, 1 always. Write a program to identify the user ID infect based on N value checked from the logs of the system

Input

The input contains the N value left by the virus and found by engineers in the logs of the system
Output
Print the userID of the infect user.

Example Input
11
Output
81

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      if (n == 1 || n == 2) { System.out.println(0); return; }
      if (n == 3) { System.out.println(1); return; }
      long a = 0, b = 0, c = 1, d = 0;
      for (int i = 4; i <= n; i++) {
         d = a + b + c;
         a = b; b = c; c = d;
      }
      System.out.println(c);
   }
}
```

Q39.Amazon Cryptography

Amazon wants to apply cryptography to its barcode scanner printed on items. Each item already has an item number, your job is to generate this cryptographed barcode number that will be printed, which should follow the following rule:

Accept a crypto key C from the user, the item number replaced is then replaced with barcode number such that each barcode digit is the Cth digit in the case if the result is less than 10. Else, a character in general alphabets post 10.

See the input and output below to understand the solution.

Input format:
First input contains the order number and the 2nd input contains the crypto key C

Output format:
Print the order number

Example Input
46734
2

Output

68956

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String order = sc.next();
        int c = sc.nextInt();
        StringBuilder sb = new StringBuilder();
        for (char ch : order.toCharArray()) {
            int val = (ch - '0') + c;
            if (val < 10) sb.append(val);
            else sb.append((char)('A' + val - 10));
        }
        System.out.println(sb.toString());
    }
}
```

Q40.Count element of string

Write a program to count the number of digits, special characters, whitespaces, and alphabets in a string. Print in count of each in same order, one per line

Input and Output
first line takes the input of a sentence
second line contains number of alphabets
third line contains number of digits
fourth line contains number of spaces
fifth line contains number of special characters

Sample
Input
Amcatuff@ #% 123

Output
Alphabets - 8
Digits - 3
Space - 2
Special Character - 3

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();
```

```
        int alpha = 0, digit = 0, space = 0, special = 0;
        for (char ch : s.toCharArray()) {
            if (Character.isLetter(ch)) alpha++;
            else if (Character.isDigit(ch)) digit++;
            else if (Character.isWhitespace(ch)) space++;
            else special++;
        }
        System.out.println("Alphabets - " + alpha);
        System.out.println("Digits - " + digit);
        System.out.println("Space - " + space);
        System.out.println("Special Character - " + special);
    }
}
```

Q41.Conditional Array
ABS School wants to provide grades to students according to their marks. There Grading Policy is as follows:

Marks
10-40 = F
41-50 = C
51-60 = B
61-80 = A
81-100 = S

All other inputs print invalid

Input -
Total number of inputs
List of marks

Output-
List of grades

Example- Input:
5
81 61 51 41 11
Output:
S A B C F

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
```

```
        for (int i = 0; i < t; i++) {
            int m = sc.nextInt();
            if (m >= 10 && m <= 40) System.out.print("F ");
            else if (m >= 41 && m <= 50) System.out.print("C ");
            else if (m >= 51 && m <= 60) System.out.print("B ");
            else if (m >= 61 && m <= 80) System.out.print("A ");
            else if (m >= 81 && m <= 100) System.out.print("S ");
            else System.out.print("invalid ");
        }
    }
}
```

Q42.Count the number of occurrence
Write a program to count the number of occurrences of string2 in string1.

Example

Input: Always Joe in Friends Joe with Joe Joe
Joe

Output: 4

Explanation:
Joe comes 4 times in a string

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s1 = sc.nextLine();
        String s2 = sc.nextLine();
        int count = 0, idx = 0;
        while ((idx = s1.indexOf(s2, idx)) != -1) {
            count++;
            idx += s2.length();
        }
        System.out.println(count);
    }
}
```

Q43.Problem Statement – Write a program to calculate the fuel consumption of your truck.The program should ask the user to enter the quantity of diesel to fill up the tank and the distance covered till the tank goes dry.Calculate the fuel consumption and display it in the format (liters per 100 kilometers).

Convert the same result to the U.S. style of miles per gallon and display the result. If the quantity or distance is zero or negative display ” is an Invalid Input”.
[Note: The US approach of fuel consumption calculation (distance / fuel) is the inverse of the European approach (fuel / distance). Also note that 1 kilometer is 0.6214 miles, and 1 liter is 0.2642 gallons.]
The result should be with two decimal place.To get two decimal place refer the below-mentioned print statement :
float cost=670.23;
System.out.printf(“You need a sum of Rs.%.2f to cover the trip”,cost);

Sample Input 1:
Enter the no of liters to fill the tank
20
Enter the distance covered
150

Sample Output 1:
Liters/100KM 13.33
Miles/gallons 17.64

Explanation:
For 150 KM fuel consumption is 20 liters,
Then for 100 KM fuel consumption would be (20/150)*100=13.33,
Distance is given in KM, we have to convert it to miles (150*0.6214)=93.21,
Fuel consumption is given in liters, we have to convert it to gallons (20*0.2642)=5.284,
Then find(miles/gallons)=(93.21/5.284)17.64

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the no of liters to fill the tank");
        double liters = sc.nextDouble();
        System.out.println("Enter the distance covered");
        double distance = sc.nextDouble();
        if (liters <= 0 || distance <= 0) {
            System.out.println("Invalid Input");
            return;
        }
        double litersPer100Km = (liters / distance) * 100;
        double miles = distance * 0.6214;
        double gallons = liters * 0.2642;
        double milesPerGallon = miles / gallons;
        System.out.printf("Liters/100KM %.2f\n", litersPer100Km);
        System.out.printf("Miles/gallons %.2f\n", milesPerGallon);
    }
}
```

Q44.Problem Statement – Vohra went to a movie with his friends in a Wave theatre and during break time he bought pizzas, puffs and cool drinks. Consider the following prices :
Rs.100/pizza
Rs.20/puffs
Rs.10/cooldrink
Generate a bill for What Vohra has bought.

Sample Input 1:
Enter the no of pizzas bought:10
Enter the no of puffs bought:12
Enter the no of cool drinks bought:5

Sample Output 1:
Bill Details
No of pizzas:10
No of puffs:12
No of cooldrinks:5
Total price=1290

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the no of pizzas bought:");
        int pizzas = sc.nextInt();
        System.out.print("Enter the no of puffs bought:");
        int puffs = sc.nextInt();
        System.out.print("Enter the no of cool drinks bought:");
        int drinks = sc.nextInt();
        int total = pizzas * 100 + puffs * 20 + drinks * 10;
        System.out.println("Bill Details");
        System.out.println("No of pizzas:" + pizzas);
        System.out.println("No of puffs:" + puffs);
        System.out.println("No of cooldrinks:" + drinks);
        System.out.println("Total price=" + total);
    }
}
```

Q45.Problem Statement – Ritik wants a magic board, which displays a character for a corresponding number for his science project. Help him to develop such an application. For example when the digits 65,66,67,68 are entered, the alphabet ABCD are to be displayed.

Assume the number of inputs should be always 4

Sample Input 1:
Enter the digits:65666768
Sample Output 1:ABCD

Explanation:
65-A
66-B
67-C
68-D

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the digits:");
        String s = sc.next();
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < 4; i++) {
            int num = Integer.parseInt(s.substring(i*2, i*2 + 2));
            sb.append((char)num);
        }
        System.out.println(sb.toString());
    }
}
```

Q46.Problem Statement – FOE college wants to recognize the department which has succeeded in getting the maximum number of placements for this academic year. The departments that have participated in the recruitment drive are CSE,ECE, MECH. Help the college find the department getting maximum placements. Check for all the possible output given in the sample snapshot
Note : If any input is negative, the output should be “Input is Invalid”. If all department has equal number of placements, the output should be “None of the department has got the highest placement”.

Sample Input 1:
Enter the no of students placed in CSE:90
Enter the no of students placed in ECE:45
Enter the no of students placed in MECH:70
Sample Output 1:
Highest placement CSE

```
import java.util.*;
public class Main {
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the no of students placed in CSE:");
        int cse = sc.nextInt();
        System.out.print("Enter the no of students placed in ECE:");
        int ece = sc.nextInt();
        System.out.print("Enter the no of students placed in MECH:");
        int mech = sc.nextInt();
        if (cse < 0 || ece < 0 || mech < 0) {
            System.out.println("Input is Invalid");
            return;
        }
        if (cse == ece && ece == mech) {
            System.out.println("None of the department has got the highest placement");
        } else if (cse > ece && cse > mech) {
            System.out.println("Highest placement CSE");
        } else if (ece > cse && ece > mech) {
            System.out.println("Highest placement ECE");
        } else if (mech > cse && mech > ece) {
            System.out.println("Highest placement MECH");
        } else {
            System.out.println("None of the department has got the highest placement");
        }
    }
}
```

Q47.Problem Statement – In a theater, there is a discount scheme announced where one gets a 10% discount on the total cost of tickets when there is a bulk booking of more than 20 tickets, and a discount of 2% on the total cost of tickets if a special coupon card is submitted. Develop a program to find the total cost as per the scheme. The cost of the k class ticket is Rs.75 and q class is Rs.150. Refreshments can also be opted by paying an additional of Rs. 50 per member.

Hint: k and q and You have to book minimum of 5 tickets and maximum of 40 at a time. If fails display “Minimum of 5 and Maximum of 40 Tickets”. If circle is given a value other than ‘k’ or ‘q’ the output should be “Invalid Input”.
The ticket cost should be printed exactly to two decimal places.

Sample Input 1:
Enter the no of ticket:35
Do you want refreshment:y
Do you have coupon code:y
Enter the circle:k

Sample Output 1:
Ticket cost:4065.25

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the no of ticket:");
        int tickets = sc.nextInt();
        if (tickets < 5 || tickets > 40) {
            System.out.println("Minimum of 5 and Maximum of 40 Tickets");
            return;
        }
        sc.nextLine();
        System.out.print("Do you want refreshment:");
        String refresh = sc.nextLine().trim().toLowerCase();
        System.out.print("Do you have coupon code:");
        String coupon = sc.nextLine().trim().toLowerCase();
        System.out.print("Enter the circle:");
        char circle = sc.nextLine().trim().toLowerCase().charAt(0);
        double ticketPrice;
        if (circle == 'k') ticketPrice = 75;
        else if (circle == 'q') ticketPrice = 150;
        else {
            System.out.println("Invalid Input");
            return;
        }
        double total = tickets * ticketPrice;
        if (tickets > 20) total *= 0.9;
        if (coupon.equals("y")) total *= 0.98;
        if (refresh.equals("y")) total += tickets * 50;
        System.out.printf("Ticket cost:%.2f\n", total);
    }
}
```


Q48.Problem Statement – Rhea Pandey's teacher has asked her to prepare well for the lesson on seasons. When her teacher tells a month, she needs to say the season corresponding to that month. Write a program to solve the above task.
Spring – March to May,
Summer – June to August,
Autumn – September to November and
Winter – December to February.

Months should be in the range 1 to 12. If not the output should be "Invalid month".

Sample Input 1:
Enter the month:11
Sample Output 1:

Season:Autumn

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the month:");
        int month = sc.nextInt();
        String season;
        if (month >= 3 && month <= 5) season = "Spring";
        else if (month >= 6 && month <= 8) season = "Summer";
        else if (month >= 9 && month <= 11) season = "Autumn";
        else if (month == 12 || month == 1 || month == 2) season = "Winter";
        else {
            System.out.println("Invalid month");
            return;
        }
        System.out.println("Season:" + season);
    }
}
```

Q49.Problem Statement – To speed up his composition of generating unpredictable rhythms, Blue Bandit wants the list of prime numbers available in a range of numbers.Can you help him out?
Write a java program to print all prime numbers in the interval [a,b] (a and b, both inclusive).
Note- Input 1 should be lesser than Input 2. Both the inputs should be positive. Range must always be greater than zero.If any of the condition mentioned above fails, then display "Provide valid input" Use a minimum of one for loop and one while loop

Sample Input 1:
2
15
Sample Output 1:
2 3 5 7 11 13

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
        if (a <= 0 || b <= 0 || a >= b) {
            System.out.println("Provide valid input");
            return;
        }
        for (int i = a; i <= b; i++) {
```

```
            if (i == 1) continue;
            int j = 2;
            boolean prime = true;
            while (j * j <= i) {
                if (i % j == 0) {
                    prime = false;
                    break;
                }
                j++;
            }
            if (prime) System.out.println(i);
        }
    }
}
```

Q50.Problem Statement – Goutam and Tanul play by telling numbers. Goutam says a number to Tanul. Tanul should first reverse the number and check if it is the same as the original. If yes, Tanul should say “Palindrome”. If not, he should say “Not a Palindrome”. If the number is negative, print “Invalid Input”. Help Tanul by writing a program.

Sample Input 1 :
21212
Sample Output 1 :
Palindrome

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int num = sc.nextInt();
        if (num < 0) {
            System.out.println("Invalid Input");
            return;
        }
        int original = num, reversed = 0;
        while (num != 0) {
            reversed = reversed * 10 + num % 10;
            num /= 10;
        }
        if (original == reversed) System.out.println("Palindrome");
        else System.out.println("Not a Palindrome");
    }
}
```

Q51.XYZ Technologies is in the process of incrementing the salary of the employees. This increment is done based on their salary and their performance appraisal rating.
If the appraisal rating is between 1 and 3, the increment is 10% of the salary.
If the appraisal rating is between 3.1 and 4, the increment is 25% of the salary. If the appraisal rating is between 4.1 and 5, the increment is 30% of the salary.
Help them to do this, by writing a program that displays the incremented salary. Write a class "IncrementCalculation.java" and write the main method in it.
Note : If either the salary is 0 or negative (or) if the appraisal rating is not in the range 1 to 5 (inclusive), then the output should be "Invalid Input".

Sample Input 1 :
Enter the salary 8000
Enter the Performance appraisal rating
3
Sample Output 1 :
8800

```
import java.util.*;
public class IncrementCalculation {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double salary = sc.nextDouble();
        double rating = sc.nextDouble();
        if (salary <= 0 || rating < 1 || rating > 5) {
            System.out.println("Invalid Input");
            return;
        }
        double increment = 0;
        if (rating >= 1 && rating <= 3) increment = 0.1 * salary;
        else if (rating > 3 && rating <= 4) increment = 0.25 * salary;
        else if (rating > 4 && rating <= 5) increment = 0.3 * salary;
        double newSalary = salary + increment;
        System.out.printf("%.2f\n", newSalary);
    }
}
```

Q52.Problem Statement : Given a string and a positive integer d. Some characters may be repeated in the given string. Rearrange characters of the given string such that the same characters become d distance away from each other. Note that there can be many possible rearrangements, the output should be one of the possible rearrangements. If no such arrangement is possible, that should also be
reported. Expected time complexity is O(n) where n is the length of the input string.

Test cases: Example 1:

Input: “abb”, d = 2
Output: “bab”

Example 2:
Input: “aacbbc”, d = 3
Output: “acbacb”

Example 3:
Input: “aaa”, d = 2
Output: Cannot be rearranged

```
import java.util.*;
public class Main {
    static class CharFreq {
        char ch;
        int freq;
        CharFreq(char ch, int freq) { this.ch = ch; this.freq = freq; }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        int d = sc.nextInt();
        if (d <= 0) { System.out.println("Cannot be rearranged"); return; }

        Map<Character, Integer> freqMap = new HashMap<>();
        for (char c : s.toCharArray()) freqMap.put(c, freqMap.getOrDefault(c, 0) + 1);

        PriorityQueue<CharFreq> pq = new PriorityQueue<>((a,b) -> b.freq - a.freq);
        for (char c : freqMap.keySet()) pq.add(new CharFreq(c, freqMap.get(c)));

        Queue<CharFreq> waitQueue = new LinkedList<>();
        StringBuilder sb = new StringBuilder();

        while (!pq.isEmpty()) {
            CharFreq cf = pq.poll();
            sb.append(cf.ch);
            cf.freq--;
            waitQueue.add(cf);
            if (waitQueue.size() >= d) {
                CharFreq front = waitQueue.poll();
                if (front.freq > 0) pq.add(front);
            }
        }

        if (sb.length() == s.length()) System.out.println(sb.toString());
        else System.out.println("Cannot be rearranged");
    }
```

}

Q53.Problem Statement –IIHM institution is offering a variety of courses to students. Students have a facility to check whether a particular course is available in the institution. Write a program to
help the institution accomplish this task. If the number is less than or equal to zero display “Invalid Range”.
Assume the maximum number of courses is 20.

Sample Input 1:
Enter no of course: 5
Enter course names:Java Oracle C++ Mysql Dotnet
Enter the course to be searched:C++

Sample Output 1:C++ course is available

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter no of course: ");
        int n = sc.nextInt();
        if (n <= 0 || n > 20) {
            System.out.println("Invalid Range");
            return;
        }
        sc.nextLine();
        System.out.print("Enter course names: ");
        String[] courses = sc.nextLine().split(" ");
        if (courses.length != n) {
            System.out.println("Invalid Range");
            return;
        }
        System.out.print("Enter the course to be searched: ");
        String search = sc.next();
        boolean found = false;
        for (String course : courses) {
            if (course.equalsIgnoreCase(search)) {
                found = true;
                break;
            }
        }
        if (found) System.out.println(search + " course is available");
        else System.out.println(search + " course is not available");
    }
```

}

Q54.Problem Statement – Mayuri buys “N” no of products from a shop. The shop offers a different percentage of discount on each item. She wants to know the item that has the minimum discount offer, so that she can avoid buying that and save money.
[Input Format: The first input refers to the no of items; the second input is the item name, price and discount percentage separated by comma(,)]
Assume the minimum discount offer is in the form of Integer.
Note: There can be more than one product with a minimum discount.

Sample Input 1:
4
mobile,10000,20
shoe,5000,10
watch,6000,15
laptop,35000,5
Sample Output 1:
shoe

Explanation: The discount on the mobile is 2000, the discount on the shoe is 500, the discount on the watch is 900 and the discount on the laptop is 1750. So the discount on the shoe is the minimum.

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine();
        double minDiscountAmount = Double.MAX_VALUE;
        List<String> minItems = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            String line = sc.nextLine();
            String[] parts = line.split(",");
            String name = parts[0].trim();
            double price = Double.parseDouble(parts[1].trim());
            double discountPercent = Double.parseDouble(parts[2].trim());
            double discountAmount = price * discountPercent / 100;
            if (discountAmount < minDiscountAmount) {
                minDiscountAmount = discountAmount;
                minItems.clear();
                minItems.add(name);
            } else if (discountAmount == minDiscountAmount) {
                minItems.add(name);
            }
```

```
        }
        for (String item : minItems) System.out.println(item);
    }
}
```

Q55.Problem Statement – Raj wants to know the maximum marks scored by him in each semester. The mark should be between 0 to 100 ,if goes beyond the range display “You have entered invalid mark.”

Sample Input 1:
Enter no of semester:3
Enter no of subjects in 1 semester:3
Enter no of subjects in 2 semester:4
Enter no of subjects in 3 semester:2
Marks obtained in semester 1: 50
60
70
Marks obtained in semester 2: 90
98
76
67
Marks obtained in semester 3: 89
76

Sample Output 1:
Maximum mark in 1 semester:70
Maximum mark in 2 semester:98
Maximum mark in 3 semester:89

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter no of semester:");
        int semesters = sc.nextInt();
        int[] subjects = new int[semesters];
        int[][] marks = new int[semesters][];

        for (int i = 0; i < semesters; i++) {
            System.out.print("Enter no of subjects in " + (i + 1) + " semester:");
            subjects[i] = sc.nextInt();
            marks[i] = new int[subjects[i]];
        }

        for (int i = 0; i < semesters; i++) {
```

```
            System.out.println("Marks obtained in semester " + (i + 1) + ":");
            for (int j = 0; j < subjects[i]; j++) {
                int mark = sc.nextInt();
                if (mark < 0 || mark > 100) {
                    System.out.println("You have entered invalid mark.");
                    return;
                }
                marks[i][j] = mark;
            }
        }

        for (int i = 0; i < semesters; i++) {
            int maxMark = -1;
            for (int j = 0; j < subjects[i]; j++) {
                if (marks[i][j] > maxMark) maxMark = marks[i][j];
            }
            System.out.println("Maximum mark in " + (i + 1) + " semester:" + maxMark);
        }
    }
}
```

Q56.Problem Statement – Bela teaches her daughter to find the factors of a given number. When she provides a number to her daughter, she should tell the factors of that number. Help her to do this, by writing a program. Write a class FindFactor.java and write the main method in it.
Note :If the input provided is negative, ignore the sign and provide the output. If the input is zero. If the input is zero the output should be “No Factors”.
Sample Input 1 :
54
Sample Output 1 :
1, 2, 3, 6, 9, 18, 27, 54

```
import java.util.*;
public class FindFactor {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int num = sc.nextInt();
        if (num == 0) {
            System.out.println("No Factors");
            return;
        }
        num = Math.abs(num);
        for (int i = 1; i <= num; i++) {
            if (num % i == 0) System.out.println(i);
```

```
    }
  }
}
```

Q57.You want to buy a particular stock at its lowest price and sell it later at its highest price. Since the stock market is unpredictable, you steal the price plans of a company for this stock for the next N days.
Find the best price you can get to buy this stock to achieve maximum profit.
Note: The initial price of the stock is 0.

Input Specification:
Input1: N, number of days
Input2: Array representing change in stock price for the day.

Output Specification:
Your function must return the best price to buy the stock at.

Example1:
Input1: 5
Input2: -39957,-17136,35466,21820,-26711
Output: -57093
Explanation: The best time to buy the stock will be on Day 2 when the price of the stock will be -57093.

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int[] changes = new int[n];
      for (int i = 0; i < n; i++) changes[i] = sc.nextInt();

      int price = 0;
      int minPrice = 0;
      for (int i = 0; i < n; i++) {
         price += changes[i];
         if (price < minPrice) minPrice = price;
      }
      System.out.println(minPrice);
   }
}
```

Q58.Given a positive whole number n, find the smallest number which has the very same digits existing in the whole number n and is greater than n. In the event that no such certain number exists, return – 1.
Note: that the returned number should fit in a 32-digit number, if there is a substantial answer however it doesn't fit in a 32-bit number, return – 1.

Example 1:
Input: 12
Output: 21
Explanation: Using the same digit as the number of permutations, the next greatest number for 12 is 21.

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        char[] digits = String.valueOf(n).toCharArray();

        int i = digits.length - 2;
        while (i >= 0 && digits[i] >= digits[i + 1]) i--;
        if (i < 0) {
            System.out.println(-1);
            return;
        }
        int j = digits.length - 1;
        while (digits[j] <= digits[i]) j--;
        char temp = digits[i];
        digits[i] = digits[j];
        digits[j] = temp;
        Arrays.sort(digits, i + 1, digits.length);
        try {
            long result = Long.parseLong(new String(digits));
            if (result > Integer.MAX_VALUE) System.out.println(-1);
            else System.out.println(result);
        } catch (NumberFormatException e) {
            System.out.println(-1);
        }
    }
}
```

Q59.Henry is extremely keen on history and every one of the ages of his family. He does a ton of exploration and understands that he has plummeted from the incomparable

Amaya line. After a ton of looking through old records and the most recent records of the general public, he can discover all the parent-kid connections in his family right from the extraordinary ruler Ming of the tradition to himself.
These connections are given in the structure of a direct exhibit where the emperor is at the main position and his kids are at pos (2i + 1) and (2i + 2)
This is the pattern followed throughout.
Henry needs to sort out every one of the kin of individual X from the information. Write a program for Henry to figure out all the siblings of person X from the data. Return the sorted list of all of Henry's siblings.
If no sibling return -1
input 1: N, the length of the array
input2: An array representing the ancestral tree
input 3: X, the person whose siblings are sought.
output – return the array of all siblings in increasingly sorted order.

Example 1 :
input 1: 5
input 2: 1,2,3,4,5
input 3: 1
output: -1
Explanation: x is the root of the tree and has no siblings

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] tree = new int[n];
        for (int i = 0; i < n; i++) tree[i] = sc.nextInt();
        int X = sc.nextInt();

        int index = -1;
        for (int i = 0; i < n; i++) {
            if (tree[i] == X) {
                index = i;
                break;
            }
        }
        if (index == 0) {
            System.out.println(-1);
            return;
        }
        int parent = (index - 1) / 2;
        List<Integer> siblings = new ArrayList<>();
        int left = 2 * parent + 1;
        int right = 2 * parent + 2;
        if (left < n && left != index) siblings.add(tree[left]);
        if (right < n && right != index) siblings.add(tree[right]);
```

```
        if (siblings.isEmpty()) System.out.println(-1);
        else {
            Collections.sort(siblings);
            for (int s : siblings) System.out.print(s + " ");
        }
    }
}
```

Q60.Problem Statement: Seema loves working with strings. She always tries to perform some operations on strings. One day she decided to find the frequencies of each of the characters in a given string but she found some difficulty in doing this so she needs your help. Help Seema by finding the frequencies of each of the characters in a given string. The input string contains only lowercase letters. The output string should contain a letter followed by its frequency, in alphabetical order (from a to z).

Input Specification:
input1: the input string

Output Specification:
Return a string representing the frequency counts of characters in the input string.

Example 1:
Input: str = “aabccccddd”
Output: a2b1c4d3

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        int[] freq = new int[26];
        for (char c : s.toCharArray()) freq[c - 'a']++;
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < 26; i++) {
            if (freq[i] > 0) sb.append((char)(i + 'a')).append(freq[i]);
        }
        System.out.println(sb.toString());
    }
}
```

Q61.Given a number n, the task is to find the remainder when n is divided by 11. The input number may be very large.

Since the given number can be very large, you can not use n % 11.

Input Specification:
inputs a large number in the form of a string

Output Specification:
Return the remainder modulo 11 of input1

Example1:
Input: str 13589234356546756
Output: 6

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String num = sc.next();
        int sum = 0;
        for (int i = 0; i < num.length(); i++) {
            int digit = num.charAt(i) - '0';
            if (i % 2 == 0) sum += digit;
            else sum -= digit;
        }
        int remainder = ((sum % 11) + 11) % 11;
        System.out.println(remainder);
    }
}
```

Q62.Problem Statement: Your house has a total of L light bulbs. Each light covers the Ai to Bi distance. Find the length covered with light.

Input Specification:

input1: L, denoting the number of light bulbs.
input2: An array of L* 2 elements. For each row i. (Ai, Bi) denote that the
light covers the distance from Ai to Bi.

Output Specification:
Your function should return the length covered by light.

Test Cases: Example 1:
input1:2
input2:{{5,10},{8, 12}}
Output: 7

Explanation:

Light 1:10 – 5 = 5 units covered.
Light 2: 12 – 8 = 4 units covered.
Common region = 10 – 8 = 2 units.
Total = 5+4 -2 = 7

```
import java.util.*;

public class Solution {
    public static int findLightCoverage(int L, int[][] intervals) {
        if (L == 0) return 0;

        Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));

        int totalLength = 0;
        int start = intervals[0][0];
        int end = intervals[0][1];

        for (int i = 1; i < L; i++) {
            if (intervals[i][0] <= end) {
                end = Math.max(end, intervals[i][1]);
            } else {
                totalLength += end - start;
                start = intervals[i][0];
                end = intervals[i][1];
            }
        }
        totalLength += end - start;

        return totalLength;
    }

    public static void main(String[] args) {
        int L = 2;
        int[][] intervals = {{5, 10}, {8, 12}};
        System.out.println(findLightCoverage(L, intervals));
    }
}
```

Q63.An astrologer gives a matrix to devilliers and tells him to add a largest row sum and largest column sum of the given matrix.The number which appears as a result is his lucky number for the final match jersey.

Write a program that adds up the largest row sum and the largest column sum from an N-rows*M-columns array of numbers to help devilliers for finding his lucky number for the final match jersey.
As a preliminary phrase , you should reformat the sequence of numbers as a matrix, whose number of rows and columns are to be specified as arguments.

Input Specification:
Input 1: Integer for row dimension of the array
Input 2: Integer for column dimension of the array
Input 3: Array elements to be entered in row major.

Output Specifications:
Largest row sum+ Largest column sum

Example 1:
Input1:4
Input2: 4
Input3: 1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4
Output: 26

```
public class Solution {
    public static int findLuckyNumber(int rows, int cols, int[] elements) {
        int[][] matrix = new int[rows][cols];
        int index = 0;

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = elements[index++];
            }
        }

        int maxRowSum = Integer.MIN_VALUE;
        for (int i = 0; i < rows; i++) {
            int rowSum = 0;
            for (int j = 0; j < cols; j++) {
                rowSum += matrix[i][j];
            }
            maxRowSum = Math.max(maxRowSum, rowSum);
        }

        int maxColSum = Integer.MIN_VALUE;
        for (int j = 0; j < cols; j++) {
            int colSum = 0;
            for (int i = 0; i < rows; i++) {
                colSum += matrix[i][j];
            }
            maxColSum = Math.max(maxColSum, colSum);
```

```
        }

        return maxRowSum + maxColSum;
    }

    public static void main(String[] args) {
        int rows = 4;
        int cols = 4;
        int[] elements = {1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4};
        System.out.println(findLuckyNumber(rows, cols, elements));
    }
}
```

Q64.Williamson is an analyst he needs to analyse a particular topic for performing analysis for that he needs to find a permutation of a given object.He don't know how to find permutation so for simplifying his work he is hiring one software developer who can code for him and find a permutation and combination of a given object.
Consider you are giving an interview to Williamson for working with him. Find a permutation of given input to proof that you are fit for his requirement.
Input Specification:
nCr where n and r are numbers given by Williamson to you nCr is defined as n! / (r! x (n-r)!) Here, n! denotes the factorial of a number. Also, you have to calculate this number as modulo

Input Specification:
Input1: The number n.
Input2: The number r.
Input3: The number m.

Output specification:
The value of nCr%m.

Example 1:
Input1: 3
Input2: 2
Input3: 10000000009
Output:3

Explanation:
n=3.r=2,m=100 So, n!3!6, r!2!2, (n-1)! 1!1. So, nCr = (6/(2*1))%1000000009 3.

```
public class Solution {
    public static long nCrModulo(int n, int r, int m) {
        if (r > n || r < 0) return 0;
        if (r == 0 || r == n) return 1;
```

```
        r = Math.min(r, n - r);

        long result = 1;
        for (int i = 0; i < r; i++) {
            result = (result * (n - i)) % m;
            result = (result * modInverse(i + 1, m)) % m;
        }

        return result;
    }

    private static long modInverse(long a, long m) {
        return power(a, m - 2, m);
    }

    private static long power(long base, long exp, long mod) {
        long result = 1;
        base = base % mod;
        while (exp > 0) {
            if (exp % 2 == 1) {
                result = (result * base) % mod;
            }
            exp = exp >> 1;
            base = (base * base) % mod;
        }
        return result;
    }

    public static void main(String[] args) {
        int n = 5;
        int r = 2;
        int m = 1000000007;
        System.out.println(nCrModulo(n, r, m));
    }
}
```

Q65.A Derangement is a permutation of n elements, such that no element appears in its original position.
For example, a derangement of 0, 1, 2, 3 is 2, 3, 1, 0.
Given a number n, find the total number of Derangements of a set of n elements.

Input Specification:
input1: N, the number of Objects
Output Specification:
Return the number of arrangements in which no object occurs at its original position
Example 1:
Input: 2

Output: 1
For two elements say 0, 1, there is only one possible derangement 1, 0

```
public class Solution {
    public static long countDerangements(int n) {
        if (n == 0) return 1;
        if (n == 1) return 0;
        if (n == 2) return 1;

        long prev2 = 1;
        long prev1 = 0;
        long current = 0;

        for (int i = 3; i <= n; i++) {
            current = (i - 1) * (prev1 + prev2);
            prev2 = prev1;
            prev1 = current;
        }

        return current;
    }

    public static void main(String[] args) {
        int n = 4;
        System.out.println(countDerangements(n));
    }
}
```

Q66.Given an n x n matrix where each of the rows and columns is sorted in ascending order, return the kth smallest element in the matrix.
Note that it is the kth smallest element in the sorted order, not the kth distinct element.

Example 1:
Input: matrix = [[1,5,9],[10,11,13],[12,13,15]], k = 8
Output: 13
Explanation: The elements in the matrix are [1,5,9,10,11,12,13,13,15], and the 8th smallest number is 13

```
import java.util.PriorityQueue;

public class KthSmallestElement {
    public static int kthSmallest(int[][] matrix, int k) {
        int n = matrix.length;
        PriorityQueue<int[]> minHeap = new PriorityQueue<>((a, b) -> a[0] - b[0]);

        for (int i = 0; i < n; i++) {
```

```
            minHeap.offer(new int[]{matrix[i][0], i, 0});
        }

        int number = 0;
        for (int i = 0; i < k; i++) {
            int[] current = minHeap.poll();
            number = current[0];
            int row = current[1];
            int col = current[2];
            if (col + 1 < n) {
                minHeap.offer(new int[]{matrix[row][col + 1], row, col + 1});
            }
        }
        return number;
    }

    public static void main(String[] args) {
        int[][] matrix = {{1, 5, 9}, {10, 11, 13}, {12, 13, 15}};
        int k = 8;
        System.out.println(kthSmallest(matrix, k));
    }
}
```

Q67.You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]).
Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return the maximum amount of water a container can store.

Example 1:
Input: height = [1,8,6,2,5,4,8,3,7]
Output: 49
Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

```
public class ContainerWithMostWater {
    public static int maxArea(int[] height) {
        int left = 0, right = height.length - 1;
        int maxArea = 0;

        while (left < right) {
            int width = right - left;
            int currentHeight = Math.min(height[left], height[right]);
            maxArea = Math.max(maxArea, width * currentHeight);
```

```
            if (height[left] < height[right]) {
                left++;
            } else {
                right--;
            }
        }
        return maxArea;
    }

    public static void main(String[] args) {
        int[] height = {1, 8, 6, 2, 5, 4, 8, 3, 7};
        System.out.println(maxArea(height));
    }
}
```

Q68.Minimum Number of Steps to Reduce X to 0
Problem: Given an array of positive integers and a target integer x , return the minimum number of steps to reduce x to exactly 0 by subtracting elements from the array, either from the beginning or the end. If it's not possible, return -1.

Input:nums=[1,1,4,2,3] ,x = 5
Output: 2
Explanation: We can subtract 3 from the end and 2 from the end to reach 0.

```
public class MinimumStepsToReduceX {
    public static int minSteps(int[] nums, int x) {
        int totalSum = 0;
        for (int num : nums) {
            totalSum += num;
        }

        int target = totalSum - x;
        if (target < 0) return -1;

        int maxLength = -1;
        int currentSum = 0;
        int left = 0;

        for (int right = 0; right < nums.length; right++) {
            currentSum += nums[right];

            while (currentSum > target && left <= right) {
                currentSum -= nums[left++];
            }
```

```
            if (currentSum == target) {
                maxLength = Math.max(maxLength, right - left + 1);
            }
        }

        return maxLength == -1 ? -1 : nums.length - maxLength;
    }

    public static void main(String[] args) {
        int[] nums = {1, 1, 4, 2, 3};
        int x = 5;
        System.out.println(minSteps(nums, x));
    }
}
```

Q69.Longest Substring Without Repeating Characters
Problem: Given a string, find the length of the longest substring without repeating characters.

Input: abcabcbb
Output:3
Explanation: The answer is "abc", with the length of 3.

```
import java.util.HashMap;

public class LongestSubstringWithoutRepeating {
    public static int lengthOfLongestSubstring(String s) {
        HashMap<Character, Integer> charIndexMap = new HashMap<>();
        int maxLength = 0;
        int left = 0;

        for (int right = 0; right < s.length(); right++) {
            char currentChar = s.charAt(right);
            if (charIndexMap.containsKey(currentChar)) {
                left = Math.max(charIndexMap.get(currentChar) + 1, left);
            }
            charIndexMap.put(currentChar, right);
            maxLength = Math.max(maxLength, right - left + 1);
        }

        return maxLength;
    }
```

```
    public static void main(String[] args) {
        String input = "abcabcbb";
        System.out.println(lengthOfLongestSubstring(input));
    }
}
```

Q70.Problem Statement: Johnny was absent in his English class when the vowel topic was taught by the teacher. His English teacher gave him two strings and told him to find the length of the longest common subsequence which contains only vowels, as a punishment for not attending English class yesterday.
Help Johnny by writing a code to find the length of the longest common subsequence.
Input Specification:

input1: First string is given by his teacher
input2: Second-string given by his teacher.
Output Specification:

Return the length of the longest common subsequence which contains only vowels.

Test Cases:
Example 1:
input1: aieef
input2: klaief
Output : 3

```
public class LongestCommonVowelSubsequence {
    public static int longestCommonVowelSubsequence(String s1, String s2) {
        String vowels = "aeiouAEIOU";
        StringBuilder vowelStr1 = new StringBuilder();
        StringBuilder vowelStr2 = new StringBuilder();

        // Extract vowels from the first string
        for (char c : s1.toCharArray()) {
            if (vowels.indexOf(c) != -1) {
                vowelStr1.append(c);
            }
        }

        // Extract vowels from the second string
        for (char c : s2.toCharArray()) {
            if (vowels.indexOf(c) != -1) {
                vowelStr2.append(c);
```

```
            }
        }

        return lcs(vowelStr1.toString(), vowelStr2.toString());
    }

    private static int lcs(String s1, String s2) {
        int m = s1.length();
        int n = s2.length();
        int[][] dp = new int[m + 1][n + 1];

        // Fill the dp array
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (s1.charAt(i - 1) == s2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1] + 1;
                } else {
                    dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
                }
            }
        }

        return dp[m][n];
    }

    public static void main(String[] args) {
        String input1 = "hello";
        String input2 = "world";
        System.out.println(longestCommonVowelSubsequence(input1, input2)); // Output: 1
(common vowel 'o')
    }
}
```