Question 1:

Problem Statement

Given N items, each with its weight and profit, and a bag with a capacity W, the task is to select items to maximize the total profit such that the sum of weights of selected items is less than or equal to the bag's capacity.

Input Format

The input consists of an integer N denoting the number of items. This is followed by N pairs of integers representing the weight and profit of each item. Finally, an integer W represents the capacity of the bag.

Integer // Number of Items

Pair of Integers // weight and profit of each item

Integer // capacity of the bag

Output Format

Print an integer representing the maximum possible profit.

Integer // Maximum possible profit

Constraints

$1 \leq N \leq 100$
$1 \leq$ Weight, Profit ≤ 1000
$1 \leq$ Bag capacity ≤ 10000

Example

Sample Input 1

```
3          // Number of Items
2 10       // Item 1 weight and profit
3 5        // Item 2 weight and profit
5 15       // Item 3 weight and profit
8          // capacity of the bag
```

Sample Output 1

```
25                  // Maximum possible profit
```

Sample Input 2

4 // **Number of Items**
1 1 // Item 1 weight and profit
3 4 // Item 2 weight and profit
4 5 // Item 3 weight and profit
5 7 // Item 4 weight and profit
7 **// capacity of the bag**

Sample Output 2

9 **// Maximum possible profit**

Sample test case Explanation

For the given input, the items are:

Item 1: Weight = 2, Profit = 10
Item 2: Weight = 3, Profit = 5
Item 3: Weight = 5, Profit = 15

The bag's capacity is 8. By selecting Item 1 and Item 3, the total weight is 7 (2 + 5) which is less than 8, and the total profit is 25 (10 + 15), which is the maximum possible.

In the **second test case**

For the given input, the items are:

Item 1: Weight = 1, Profit = 1
Item 2: Weight = 3, Profit = 4
Item 3: Weight = 4, Profit = 5
Item 4: Weight = 5, Profit = 7

The bag's capacity is 7. By selecting Item 2 and Item 3, the total weight is 7 (2 + 5) which is equal to 7, and the total profit is 9 (4 + 5), which is the maximum possible.

Question 2: Password of the treasure box

Problem Statement

In a whimsical forest, a special tree called the AVL Tree was home to magical creatures like a wise owl, a swift squirrel, a graceful deer, a playful rabbit, and a majestic unicorn. Picture yourself strolling through this enchanted place and observing these creatures joining the tree. Now, I'd like you to share the order in which you encounter them, mentioning their unique energies, just like the magic they bring. Let me know the sequence you witness as you explore the forest – specifically, I'm interested in the output of the AVL Tree in preorder traversal after all the magical creatures have become a part of it.

Input Format

The input begins with an integer N denoting the number of nodes to be inserted into the AVL tree. This is followed by N integers representing the values of the nodes to be inserted.

Output Format

Print the preorder traversal of the created AVL Tree.

Constraints :
$1 \leq N \leq 1000$
Nodes' values are integers

Example
Input:
5 // number of nodes
30 20 40 10 50 // values of nodes to be inserted

Output:
30 20 10 40 50

Explanation:
For the given input, nodes 30 20 40 10 50 are inserted into the AVL tree, and the preorder traversal of the created AVL tree is 30 20 10 40 50.

3.

Problem Statement
Write a Java code to perform the Node Deletion of the Leaf Nodes form the BST. User enters the value of the nodes.

Your task is to find the Left-Most Leaf Child & Right-Most Leaf child in the BST, delete them, and then print the POST Order traversal of that given Tree.

Input Format

An array of integers (7 values)

Output Format

Post-order traversal of tree after applying deletion operation

Note: Integer array length must be equal to 7.

Example

Sample Input 1

10 5 3 7 20 18 25

Sample Output 1

5 20 10

4. Robot Task Scheduling

Problem Statement 4

You are working on a project that involves designing a dynamic programming algorithm for a robotic system. The robot has to navigate through a grid, and at each cell, there is a task to be completed. Each task has a specific duration and a corresponding profit.

The robot can move only right or down in the grid, and it cannot revisit a cell. Your goal is to maximize the total profit the robot can achieve by scheduling tasks along its path.

Write a Java program using dynamic programming (tabulation) to find the maximum total profit. The input will consist of the dimensions of the grid (rows (R) and columns (C)) and the duration-profit values for each cell. You need to output the maximum total profit achievable by the robot.

Input Constraints :

(1 <= R, C <= 100).

Input Format

The first line contains two space-separated integers: the number of rows (**R**) and the number of columns (**C**) in the grid (1 <= R, C <= 100).

The next **R** lines contain **C** pairs of space-separated integers each: the duration and profit of each task at the corresponding cell.

Output Format

Print the maximum total profit achievable by the robot.

Example

Sample Input 1

3 4 // number of rows (**R**) and the number of columns (**C**)
2 5 1 3 //Grid Values
3 7 2 8
4 8 3 4

Sample Output 1

29 // maximum total profit achievable by the robot

Sample Input 2

2 2 // number of rows (**R**) and the number of columns (**C**)
14 56 //Grid Values
52 9

Sample Output 2

79

Sample test case Explanation

In the first test case, the optimal path for the robot is (1,1) -> (1,2) -> (2,2) -> (2,3) -> (3,3) -> (3,4), maximizing the total profit.

In the second test case, the optimal path for the robot is (1,1) -> (1,2) -> (2,2) maximizing the total profit.

5. **Decode Ways**

Problem Statement
A message containing letters from A-Z can be encoded into numbers using the following mapping:

'A' -> "1"

'B' -> "2"

...

'Z' -> "26"

To decode an encoded message, all the digits must be grouped and then mapped back into letters using the reverse of the mapping above (there may be multiple ways).

Input Format

contains an input string.

Output Format

An integer. (Total number of ways the code can be decoded)

Constraint

- 1 <= s.length <= 100
- s contains only digits and may contain leading zero(s).

Example

Sample Input 1

11106 // (encoded message string)

Sample Output 1

2 // number of ways the code can be decoded

Sample Input 2

226 // (encoded message string)

Sample Output 2

3 // number of ways the code can be decoded

Sample test case Explanation

In the first test case, "11106" can be mapped into:

"AAJF" with the grouping (1 1 10 6)

"KJF" with the grouping (11 10 6)

Note that the grouping (1 11 06) is invalid because "06" cannot be mapped into 'F' since "6" is different from "06".

So, the total number of ways is 2.

In the second test case, “226” can be mapped into:

“BZ” with the grouping (2 26)

“VF” with the grouping (22 6)

“BBF” with the grouping (2 2 6)

So, the total number of ways is 3.

6. **lowest common ancestor (LCA)**

Problem Statement

Given a binary search tree (BST), find the lowest common ancestor (LCA) node of two given nodes in the BST.
According to the definition of LCA on Wikipedia: "The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow **a node to be a descendant of itself**)."

Constraints:

- The number of nodes in the tree is in the range [2, 10^5].
- -10^9 <= Node.val <= 10^9
- All Node.val are **unique**.
- p != q
- p and q will exist in the BST.

Input Format

First Line: An integer array separated by space

Second Line: An integer (value of P)

Third line: An integer (value of q)

Output Format

An Integer.

Example

Sample Input 1

6 2 8 0 4 7 9 3 5 // (Integer Array)

2 // (Value of P)

8 // value of q

Sample Output 1

6

Q 7. **Crack the Passcode**

Problem Statement 7
Rahul, Raj and their temporary buddies suffix and prefix found the harmony temple. Its doors are temporarily locked and no one was able to open the door.

Later they found a string S written on the rock and spoke the string aloud but nothing happened. Then Raj thought the password might be substring t of the string S.
Prefix supposed that the substring t is the beginning of the string S, Suffix supposed that the substring t should be the end of the string S and Raj supposed that t should be located somewhere inside the string S, that is, t is neither its beginning nor its end.
Rahul chose the substring t so as to please all his companions. Besides, from all the acceptable variants, Rahul chose the longest one(as he loves long strings). When Rahul spoke aloud, doors suddenly opened.
You are given the string S. Find the substring t or determine that such substring does not exist.

Input format
The first line contains the non-empty string S

Output format
Print the string t. If a suitable t string does not exist, then print "Not so Cool" without the quotes.

Constraints
1 <=S <= 10^6

Time Limit
1 second

Example
Sample Input
qwertyqwertyqwerty

Sample Output
qwerty

Sample Test Case Explanation
qwerty is the longest substring present in the qwertyqwertyqwerty satisfying the conditions.

Default Code

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

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

```
    //write your code here

            }
}
```

Que 8. **Hearing Challenge**

Problem Statement

You talked to Prepbuddy and asked him a question. You know that when he wants to answer "Yes", he repeats “Yes” many times in a row.

Because of the noise, you only heard part of the answer — some substring of it. That is, if he answered YesYes, then you could hear esY, YesYes, sYes, e, but you couldn't Yess, YES or se.

Determine if it is true that the given string S is a substring of YesYesYes... (Yes repeated many times in a row)

Input format

The first line of input data contains the singular T — the number of test cases.

Each test case is described by a single string of letters S — the part of the Prepbuddy answer that you heard.

Output format

Print "YES" if the specified string s is a substring of the string YesYesYes...Yes (the number of words Yes is arbitrary) and "NO" otherwise(without quotes).

Constraints

1 <= T <= 100

1 <= S <= 50

Time Limit

1 second

Example

Sample Input

5

YES

esYes

prepbytes

es

se

Sample Output
NO
YES
NO
YES
NO

Sample Test Case Explanation
In the given examples , **esYes** and **es** are the given substring of YesYesYes(Yes repeated many times in a row)

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

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

    //write your code here

            }
}
```

Que 9 **:-** Given an array and a positive integer k, find the first negative integer for each window(contiguous subarray) of size k. If a window does not contain a negative integer, then print 0 for that window.

Input : arr[] = {12, -1, -7, 8, -15, 30, 16, 28} , k = 3
Output : -1 -1 -7 -15 -15 0

Example :-

Input:
5
-8 2 3 -6 10
2
Output:
-8 0 -6 -6
Explanation:
First negative integer for each window of size k
{-8, 2} = -8
{2, 3} = 0 (does not contain a negative integer)

{3, -6} = -6
{-6, 10} = -6

Que 10. Longest Increasing Subsequence

Problem Statement : Given an array of integers, find the length of the longest subsequence in the array such that all elements of the subsequence are sorted in strictly increasing order.

Input Format : The input begins with an integer N denoting the size of the array. This is followed by N integers representing the elements of the array.

Output Format: Print an integer representing the length of the longest increasing subsequence.

Constraints :
$1 \leq N \leq 1000$
$-10^9 \leq$ Array elements $\leq 10^9$

Time Limit : 1 second

Example
Input:
8
10 22 9 33 21 50 41 60

Output:
5

Explanation:
For the given input array [10, 22, 9, 33, 21, 50, 41, 60], the longest increasing subsequence is [10, 22, 33, 50, 60] with a length of 5.

Default Code :

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

}

Coding Block Link: https://ide.codingblocks.com/s/681166

Question 11:

Problem Statement

You are given an integer array “heights” representing the heights of buildings, some bricks, and some ladders. You start your journey from building 0 and move to the next building by possibly using bricks or ladders. While moving from building i to building i+1 (0-indexed), If the current building's height is greater than or equal to the next building's height, you do not need a ladder or bricks. If the current building's height is less than the next building's height, you can either use one ladder or (h[i+1] - h[i]) bricks. If the number of bricks is less than the difference in the two building heights then you cannot move to the next building. Return the furthest building index (starting from the 0-indexed array) you can reach if you use the given ladders and bricks optimally.

Note: You can use “PriorityQueue” in java.

Input Format

Integer array

Integer // Number of Bricks

Integer // Number of Ladders

Output Format

Integer // Furthest Building Index

Constraints

1 <= heights.length <= 10^5

1 <= heights[i] <= 10^6

0 <= bricks <= 10^9

0 <= ladders <= heights.length

Example

Sample Input 1

4 2 7 6 9 14 12 // Integer array without brackets and separated by space

5 // Number of Bricks

1 // Number of Ladders

Sample Output 1

4 // Furthest Building Index

Sample Input 2

14 3 19 3 // Integer array without brackets and separated by space

14 // Number of Bricks

0 // Number of Ladders

Sample Output 2

1

Sample test case Explanation

In the first test case, starting at building 0, you can follow these steps:

- Go to building 1 without using ladders or bricks since 4 >= 2.

- Go to building 2 using 5 bricks. You must use either bricks or ladders because 2 < 7.

- Go to building 3 without using ladders or bricks since 7 >= 6.

- Go to building 4 using your only ladder. You must use ladders because 6 < 9 and you have 0 bricks left.

It is impossible to go beyond building 4 because you do not have any more bricks or ladders.

In the **second test case** starting at building 0, you can follow these steps:

- Go to building 1 without using ladders or bricks since 14 >= 3.

It is impossible to go beyond building 1 because you do not have any enough bricks or ladders.

Question 12: Password of the treasure box

Problem Statement

While digging in his backyard, Anil has found a treasure box and a bottle with a note inside it. The first line in the note says," Biggers will be rewarded and the Smalls will be slaughtered" and the second line is some arbitrary space separated words which does not make any sense. After

searching about such a treasure and note online, he came to understand that the note holds the password for the treasure box. The arbitrary space separated words actually form a number which could be used to open the box.

The number of those arbitrary words in the note is the number of digits of the required number to open the box. A word represents a digit that cannot be less than 0 and greater than 9. **So, if the word deciphered into a number comes greater than 9, it is taken as 9 and if it comes less than 0, it is taken as 0**. The word can be transformed into a number by using the following rules:

1. Alphabets written in Capital are to be added while small ones are to be subtracted.
   Example:
   If the word is ANil
   Value of A and N is to be added and the value of i and l is to be subtracted.
2. Alphabets **(either small case or upper case)** are assigned values in ascending order from 1 to 26. Example:
   A-1, C-3……Z-26
   So word ANil will give 0.

Help Anil write a JAVA program to find the code to open the treasure box and see what is inside it.

Input Format

String: Space Separated Words

DO NOT INPUT numbers or alphanumeric characters.

Output Format

Returns the code Number

Example

Sample Input 1

ABc CDeF GHi

Sample Output 1

086

Sample Input 2

ANil

Sample Output 2

0

Sample test case Explanation

In the first test case, there are 3 words

for the first word ABc – A-1, B-2, c-3 (1+2 -3 =0)

for the second word CDeF – C-3, D-4, e-5, F-6 (3 + 4 -5 +6 =8)

for the third word GHi – G-7, H-8, i-9 (7 +8 -9 =6)

So, the code is 086

In the second test case, there is single word

For ANil – A-1, N-14, i-9, l-12 (1 +14 -9-12 =-6 that automatically converts to 0)

As if the number is less than 0 (zero) then the code is equal to 0.

13. **Making students list by their roll numbers and names and search particular student by roll no.**

Problem Statement 13

Amrit is teaching the ADI subject in the class. He said to all students to come one by one and tell their names. Amrit is making the list of students in a sheet and adding roll no by default from 1,2,3 … and so on. After that, he also wants to search for the student, by giving the roll number and checking whether present or not.

Create a code for Amrit to input the names of the students, and their roll numbers should be added by default from 1 2 3 ….so on. After that input a roll number for searching, whether it is present or not. Print the complete list of the students with their roll numbers into descending order according to roll numbers, and check whether the searching roll no is present or not.

Input Format

First Line: Space Separated Words consists of input strings containing the names of students.

Second Line: Input the Roll no. of the student that needs to be checked present or not.

Output Format

Print the list of students with names and Roll numbers (in descending order)

Next Line returns the status of the student based on the roll number input (Present or not present)

Note: You can use "**Hashtable**" in java for making the students list with their roll numbers.

Example

Sample Input 1

```
Amit Sumit Anil    // First Line list of student names separated by space
4                  // Roll number of the student that needs to be checked (present or not)
```

Sample Output 1

```
3 Anil
2 Sumit
1 Amit
not present
```

Sample Input 2

```
Sumit Seema Gauri          // First Line list of student names separated by space
1                  // Roll number of the student that needs to be checked (present or not)
```

Sample Output 2

```
3 Gauri
2 Seema
1 Sumit
present
```

Sample test case Explanation

In the first test case, there are 3 student names taking the first word and assigning the roll number 1 to that student, similarly taking the next word and assigning the roll number 2 to that student and repeating the same process till the end of the input list. The second input is the roll number which is 4 which checks in the student list if the roll number is present, it prints present otherwise it prints not present, 4 is not present so it prints not present.

In the second test case, there are 3 student names taking the first word and assigning the roll number 1 to that student, similarly taking the next word and assigning the roll number 2 to that student and repeating the same process till the end of the input list. The second input is the roll number which is 1 which checks in the student list if the roll number is present, it prints present otherwise it prints not present, 1 is present in the list so it prints present.

14.

Problem Statement

Given an integer array A of size N. You need to count the number of special elements in the given array. An element is special if the removal of that element makes the array balanced.

The array will be balanced if the sum of the even index element is equal to the sum of the odd index element.

Input Constraints :

1 <= N <= 10^5

1 <= A[i] <= 10^9

Input Format

Only Single Input consists of an integer array A of size N. Integer array consists of integers separated by space.

Output Format

Returns an integer denoting the count of special elements

Example

Sample Input 1

2 1 6 4 // Single input integer array (Integers separated by space)

Sample Output 1

1 // Integer denoting count of special elements

Sample Input 2

5 5 2 5 8 // Single input integer array (Integers separated by space)

Sample Output 2

2

Sample test case Explanation

In the first test case, there are 4 elements in the array and if I remove the element at index 1 i.e., 1 then the remaining 3 elements are 2 6 4 (sum of elements at odd index and the sum of element at even index is equal) so special element count is 1 as after removing any other element the sum is not equal.

In the second test case, there are 5 elements in the array. Taking each one by one each element and checking the sum of elements of odd and even indexes, if the sum is equal then consider that element as a special element otherwise not. After removing element at index zero i.e., 5 check the sum is equal (5+5 = 2+8) so 5 is special element. Similarly for element at array index 1 check the sum is equal (5+5 = 2+8) so 5 is special element. Repeat the above process for all the elements and at last count all the special elements and print in this test case the total number of special elements are 2.

15. **Detecting Plagiarism in Academic Papers**

Problem Statement
In an academic institution, the faculty is concerned about the possibility of plagiarism in student papers. To address this concern, the institution has decided to implement a plagiarism detection system. The system aims to find instances where a specific pattern (a sequence of words or phrases) appears in multiple papers. As part of this system, you are tasked with implementing a solution that efficiently identifies the count of occurrences of a given pattern within a large collection of text documents.

Input Format

Line1: Collection of academic papers stored as a text document
Line 2: Specific pattern that you want to search for within these papers.

Output Format

The expected output of the program would be the count of occurrences of the pattern in the text document.

Example

Sample Input 1

This is an academic paper discussing the importance of algorithms in computer science. algorithms play a crucial role in various applications, including plagiarism detection. // First input text (separated by space)
algorithm // Second input pattern that needs to be checked (case sensitive)

Sample Output 1

2 // count of occurrences of the second input in the first input

Sample Input 2

Java is a powerful programming language commonly used in software development. // First input text (separated by space)
java // Second input pattern that needs to be checked (case sensitive)

Sample Output 2

0 **// count of occurrences of the second input in the first input**

Sample test case Explanation

In the first test case, count the number of occurrences in first input. algorithm is present two times so the output is 2.

In the second test case, as its case sensitive java is not present in text anywhere, so its output is 0.

16. **The Lost Palindrome Stones**

Problem Statement

Once upon a time, in the mystical land of Enchantoria, there existed a hidden treasure known as the "Palindrome Stones." Legend had it that these magical stones, when placed together in a specific sequence, had the power to unlock an ancient portal to a realm of unimaginable wonders.

The wise guardian of Enchantoria, an ancient wizard named Eldor, had inscribed a prophecy that foretold the discovery of the Palindrome Stones by a chosen one. The prophecy revealed that these magical stones were embedded within a cryptic pattern, and only those who could decipher the pattern and extract the palindrome substrings would unveil the path to the hidden treasure.

Your quest begins as you, the chosen one, embark on a journey to decipher the ancient pattern. Eldor provides you with a magical scroll containing the mysterious pattern. The pattern is a sequence of symbols, each holding the potential to reveal a part of the palindrome sequence.

Your task is to write a program that extracts all possible palindrome substrings from the given pattern and returns the indices where each of these substrings is located. The indices will guide you to arrange the Palindrome Stones in the correct order, unlocking the portal to the realm of wonders.

Each palindrome substring discovered represents a fragment of the ancient incantation needed to activate the portal. Eldor, in his wisdom, has left clues within the pattern, and by identifying these palindromes and their positions, you will uncover the hidden magic within the stones.

As you delve into the enchanting world of Enchantoria, remember that the indices of the palindrome substrings are the key to your success. Write a program to fulfill the prophecy, unravel the secrets of the Palindrome Stones, and open the portal to the extraordinary realm that awaits you. The fate of Enchantoria rests in your hands. Good luck, brave adventurer!

Input Format

Any string pattern

Output Format

The program prints the palindrome substrings (**minimum of length 3**) found in the pattern along with their indices.

Example

Sample Input 1

racecar // any string pattern

Sample Output 1

cec 2-4 // output palindrome substring of min length 3 along with indices
aceca 1-5
racecar 0-6

Sample Input 2

enchantoria

Sample Output 2

No Palindrome string found

Sample test case Explanation

In the first test case, output palindrome substring of min length 3 along with indices.

In the second test case, there is no palindrome substring so print “No Palindrome string found”

Question 17: Difference in Ascii Codes

Problem Statement

Take as input S, a string. Write a program that inserts between each pair of characters the difference between their ascii codes and print the answer.

Input Format

String: The Input String S can be alphabets, numbers or alphanumeric characters.

Output Format

String: String with difference between characters

Example

Sample Input 1

acb

Sample Output 1

a2c-1b

Sample Input 2

A12$

Sample Output 2

A-16112-14$

Question 18: Ugly Number

Problem Statement: Input an integer n and check whether n is an ugly number or not.Return true if n is an ugly number otherwise false.An ugly number is a positive integer whose prime factors are limited to 2, 3, and 5.

Input Format

Integer: Only integer. DO NOT INPUT string, floating point numbers or any other data type.

Output Format

Boolean: For each test case, return true or false.

Constraints

-2^31 <= n <= 2^31 - 1

Example

Sample Input 1

6

Sample Output 1

true

Sample Input 2

34

Sample Output 2

false

Question 19: Reducing the Target number to Zero

Problem Statement

Given an integer “target”, return the number of steps to reduce it to zero.In one step, if the current number is even, you have to divide it by 2, otherwise, you have to subtract 1 from it.

Input Format

Integer: Only integer. DO NOT INPUT string, floating point numbers or any other data type.

Output Format

Integer: Number of steps required for reduction of Target to 0.

Constraints

0 <= num <= 10^6

Example

Sample Input 1

14

Sample Output 1

6

Sample Input 2

23

Sample Output 2

8

Question 20: Odd Even character

Problem Statement: Take as input S, a string. Write a function that replaces every even character with the character having just higher ASCII code and every odd character with the character having just lower ASCII code. Print the value returned.

Input Format

String: Can be a combination of alphabets(Both uppercase and lowercase),numbers, special characters etc.

Output Format

String: Required answer

Example

Sample Input 1

abcg

Sample Output 1

badf

Sample Input 2

ABcdfg

Sample Output 2

BAdcgf

Question 21: Symmetric Numbers

Problem Statement: Take two positive integers low and high for input.An integer x consisting of 2 * n digits is symmetric if the sum of the first n digits of x is equal to the sum of the last n digits of x. Numbers with an odd number of digits are never symmetric.

Return the number of symmetric integers in the range [low, high].

Input Format

Integer: Low and High, Only integers.

Output Format

Integer: Count of Symmetric Numbers.

Constraints

1 <= low <= high <= 10^4

Example

Sample Input 1

1

100

Sample Output 1

9

Sample Input 2

23

900

Sample Output 2

7

Question 22: Count Ways to Cover Distance

Problem Statement : Given a distance 'dist', count the total number of ways to cover the distance using 1, 2, or 3 steps at a time.

Input Format
Integer: The input consists of a single integer representing the distance 'dist'.

Output Format
Integer: Return an integer representing the total number of ways to cover the distance.

Constraints
The distance 'dist' is a non-negative integer.
0 <= dist <= 10^5

Example
Sample Input 1
4

Sample Output 1
7

Sample Input 2
14

Sample Output 2
3136

Question 23: Non-overlapping Intervals

Problem Statement : Given an array of intervals, intervals where intervals[i] = [start, end], return the minimum number of intervals you need to remove to make the rest of the intervals non-overlapping.

Input Format
Integer: The input begins with an integer N denoting the number of intervals. This is followed by N pairs of integers representing the start and end points of intervals.

Output Format

Print an integer representing the minimum number of intervals to be removed.

Constraints :

$1 \leq N \leq 1000$

$-10^6 \leq$ Start and End points $\leq 10^6$

Example

Sample Input 1

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

Sample Output 1

```
1
```

Sample Input 2

```
2
1 2
2 3
```

Sample Output 2

```
0
```

Default Code

```
import java.util.*;

class Main {
   static int eraseOverlapIntervals(int[][] intervals) {
      // Enter your code here
   }

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

      int n = scanner.nextInt();
      int[][] intervals = new int[n][2];

      for (int i = 0; i < n; i++) {
         intervals[i][0] = scanner.nextInt();
         intervals[i][1] = scanner.nextInt();
      }
```

```
        System.out.println(eraseOverlapIntervals(intervals));

        scanner.close();
    }
}
```

Question 24: Left View of Binary Search Tree (BST)

Problem Statement : Given a Binary Search Tree (BST), find the left view of the tree, which contains all the nodes visible from the left side. The left view of a BST is the set of nodes visible when the tree is viewed from the left-hand side.

Input Format
The input begins with an integer n representing the number of nodes in the BST. This is followed by n integers denoting the values of the nodes in the BST.

Output Format
Print the values of nodes that are visible from the left side of the BST.

Constraints
1 ≤ Number of nodes ≤ 10^4
Values of nodes are unique integers within the range [-10^4, 10^4].

Example

Sample Input 1

7

20 9 25 5 12 22 30

Sample Output 1

20 9 5

Sample Input 2

6

40 20 60 10 30 50

Sample Output 2

40 20 10

Default Code

```
import java.util.*;

class Main {
   class Node {
      int data;
      Node left, right;

      Node(int item) {
         data = item;
         left = right = null;
      }
   }

   Node root;

   Main() {
      root = null;
   }

   void insert(int key) {
      root = insertRec(root, key);
   }

   Node insertRec(Node root, int key) {
      if (root == null) {
         root = new Node(key);
         return root;
      }

      if (key < root.data)
         root.left = insertRec(root.left, key);
      else if (key > root.data)
         root.right = insertRec(root.right, key);
```

```
        return root;
    }

    void leftView() {
        leftViewUtil(root, 1);
    }

    void leftViewUtil(Node node, int level) {
      // Write code here
    }

    int maxLevel = 0;

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

        int numNodes = scanner.nextInt();

        for (int i = 0; i < numNodes; i++) {
            int value = scanner.nextInt();
            tree.insert(value);
        }

        tree.leftView();

        scanner.close();
    }
}
```

Question 25: Word Segmentation using Dictionary

Problem Statement : Given an input string and a dictionary of words, determine if the input string can be segmented into a space-separated sequence of dictionary words.

Input Format

The input consists of a string s followed by an integer n denoting the number of words in the dictionary. This is followed by n strings representing the words in the dictionary.

Output Format

Print "Yes" if the input string can be segmented into words from the dictionary; otherwise, print "No".

Constraints :

1 ≤ Length of string, Length of dictionary words ≤ 1000

1 ≤ Number of words in the dictionary ≤ 100

Example

Sample Input 1

applepie
5
apple
pie
applep
ap
P

Sample Output 1

Yes

Sample Input 2

cars
1
car

Sample Output 2

No

Default Code

```
import java.util.*;

class Main {

   static boolean wordBreak(String s, List<String> wordDict) {
      // Enter code here
   }

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

      String s = scanner.next();
```

```
        int n = scanner.nextInt();
        List<String> wordDict = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            wordDict.add(scanner.next());
        }

        if (wordBreak(s, wordDict)) {
            System.out.println("Yes");
        } else {
            System.out.println("No");
        }

        scanner.close();
    }
}
```

Question 26: Postorder Traversal of Binary Search Tree

Problem Statement : Given a Binary Search Tree (BST), the task is to perform a postorder traversal of the tree and print the nodes' values. Also implement a method for performing the insertion operation into the BST

Input Format
The input begins with an integer N denoting the number of nodes in the BST. This is followed by N integers representing the values to be inserted into the BST.

Output Format
Print the nodes' values in postorder traversal separated by spaces.

Constraints :
$1 \leq N \leq 1000$
Nodes' values are integers

Example
Sample Input 1
7
50 30 70 20 40 60 80

Sample Output 1

20 40 30 60 80 70 50

Sample Input 2

4

8 4 12 6

Sample Output 2

6 4 12 8

Default Code

```
import java.util.*;

class Node {
   int data;
   Node left, right;

   Node(int value) {
      data = value;
      left = right = null;
   }
}

class Main {
   Node root;

   Main() {
      root = null;
   }

   void insert(int value) {
      // Enter Code here
   }



   void postorderTraversal(Node node) {
```

```
        // Enter Code here
    }

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


        int n = scanner.nextInt();

        Main tree = new Main();
        for (int i = 0; i < n; i++) {
            int value = scanner.nextInt();
            tree.insert(value);
        }



        scanner.close();
    }
}
```

Question 27: Abhishek and Prefix

Problem Statement

Abhishek loves the string algorithms very much and his teacher gave him a task in which he was provided with two strings S and T and he had to find the maximum length of some prefix of the string S which occurs in string T as a subsequence.

As Abhishek is having an exam tomorrow and he isn't having time, he wants your help with the solution.

Input format

The first line represents the string S.

The second line contains the string T.

Output format

Print the required answer.

Constraints

1 <= S, T <= 10^6

Note: Both strings consist of lowercase Latin letters.

Example
Sample Input 1
digger
biggerdiagram

Sample Output 2
3
Sample Input 2
absolute
Itisabsentinthedrive

Sample Output 2
3

Question 28: Maximizing Jewelry Value

Problem Statement: In a small village renowned for its skilled artisans, Arjun stands out as a master jeweler. His exquisite collections of necklaces, bracelets, and earrings have always been the talk of the town. As the village fair approaches, Arjun plans to exhibit his three most prized collections: a collection of elegant necklaces valued at 60 gold coins and weighing 10 kilograms in total, a set of exquisite bracelets worth 100 gold coins with a collective weight of 20 kilograms, and a range of delicate earrings priced at 120 gold coins, weighing a total of 30 kilograms.

However, Arjun faces a unique challenge. His carriage, pulled by his trusty horse, has a variable weight limit, which is equal to input weight. Arjun knew he had to choose wisely to maximize the value of the jewelry he could transport. He pondered over how he could maximize the profit while adhering to the weight constraint.
Your task is to help Arjun decide how to carry the jewelry. You can suggest taking a full item or a fraction of it, keeping in mind the weight limit. The objective is to maximize the total value of the jewelry that Arjun can carry to the fair.

Input format
Integer: Only integer. DO NOT INPUT string, floating point numbers or any other data type.

Output format
Double: Print the required answer

Example
Sample Input 1

50

Sample Output 2
240.0

Sample Input 2
75

Sample Output 2
280.0

Question 29: Path Sum of Binary Search Tree

Problem Statement : Given a pre-constructed BST and a target sum, determine if there exists a path from the root to any leaf whose values sum up to the target. You need to only implement the hasPathSum(node, int) method.

Input Format
Integer: Array size, array elements, target sum. Only the hasPathSum() method is to be implemented

Output Format
Boolean: Return a boolean value:
true, if there exists a path with the given sum
false, otherwise

Constraints
Values of nodes are integers.
The array is not sorted.
Tree nodes are not null.

Example

Sample Input 1
5
1 2 3 4 5
15

Sample Output 1
true

Sample Input 2
7
3 6 1 4 7 2 9
11

Sample Output 2
false

Default Code

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

class Main {

   static class node {
      int key;
      node left, right;
   };

   static node newNode(int item) {
      node temp = new node();
      temp.key = item;
      temp.left = temp.right = null;
      return temp;
   }

   static node insert(node node, int key) {

      if (node == null)
         return newNode(key);

      if (key < node.key) {
         node.left = insert(node.left, key);
      } else if (key > node.key) {
         node.right = insert(node.right, key);
      }

      return node;
   }
```

```
    static boolean hasPathSum(node root, int targetSum) {
        // Enter your code here
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        node root = null;
        while (N != 0) {
            root = insert(root, sc.nextInt());
            N--;
        }
        int targetSum = sc.nextInt();
        System.out.println(hasPathSum(root, targetSum));
    }
}
```

Question 30: Inorder Successor in a BST

Problem Statement : Given a Binary Search Tree (BST) and a reference to a Node x in the BST, write a Java program to find the Inorder Successor of the given node in the BST.

Input Format
The input begins with an integer n representing the number of nodes in the BST. This is followed by n integers denoting the values of the nodes in the BST. Then, an integer ref is provided as the value of the reference node x.

Output Format
Print the value of the Inorder Successor node's data or "No inorder successor found" if there's no successor.

Constraints
1 ≤ Number of nodes ≤ 10^5
Values of nodes are unique integers within the range [-10^5, 10^5].

Example
Sample Input 1
5
50 30 70 20 40

30

Sample Output 1
40

Sample Input 2
6
40 20 60 10 30 50
30

Sample Output 2
40

Default Code

```
import java.util.*;

class Main {

   class Node {
      int data;
      Node left, right;

      Node(int item) {
         data = item;
         left = right = null;
      }
   }

   Node root;

   Main() {
      root = null;
   }

   void insert(int key) {
      root = insertRec(root, key);
   }
```

```
Node insertRec(Node root, int key) {
   if (root == null) {
      root = new Node(key);
      return root;
   }

   if (key < root.data)
      root.left = insertRec(root.left, key);
   else if (key > root.data)
      root.right = insertRec(root.right, key);

   return root;
}

Node inorderSuccessor(Node root, Node x) {
  // Enter your Code here
}

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

   int numNodes = scanner.nextInt();

   for (int i = 0; i < numNodes; i++) {
      int value = scanner.nextInt();
      tree.insert(value);
   }

   int referenceValue = scanner.nextInt();
   Node x = tree.findNode(tree.root, referenceValue);

   Node result = tree.inorderSuccessor(tree.root, x);
   if (result != null)
      System.out.println(result.data);
   else
      System.out.println("No inorder successor found");

   scanner.close();
}
```

```
    Node findNode(Node root, int value) {
        if (root == null || root.data == value)
            return root;

        if (value < root.data)
            return findNode(root.left, value);
        else
            return findNode(root.right, value);
    }
}
```

Question 31: Find Diameter of a Binary Search Tree

Problem Statement : Given a Binary Search Tree (BST), determine its diameter - the length of the longest path between any two nodes in the tree. The path may or may not pass through the root.

Input Format
The input begins with an integer N denoting the number of nodes in the BST. This is followed by N integers representing the values to be inserted into the BST.

Output Format
Print an integer representing the diameter of the BST.

Constraints
$1 \leq N \leq 1000$
Nodes' values are integers

Example
Sample Input 1
7
50 30 70 20 40 60 80

Sample Output 1
5

Sample Input 2
3
50 40 30

Sample Output 2
3

Explanation:
For the given sample input 1, nodes 50 30 70 20 40 60 80 are inserted into the BST. The diameter of the BST is 6, representing the longest path between nodes in the tree.

Default Code

```
import java.util.*;

class Node {
   int data;
   Node left, right;

   Node(int value) {
      data = value;
      left = right = null;
   }
}

class Main {
   Node root;

   Main() {
      root = null;
   }

   void insert(int value) {
      root = insertRec(root, value);
   }

   Node insertRec(Node root, int value) {
      if (root == null)
         return new Node(value);

      if (value < root.data)
         root.left = insertRec(root.left, value);
      else if (value > root.data)
         root.right = insertRec(root.right, value);
```

```
        return root;
    }

    int height(Node node) {
        // Enter code here
    }

    int diameter(Node root) {
        // Enter code here
    }

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

        int n = scanner.nextInt();

        Main tree = new Main();
        for (int i = 0; i < n; i++) {
            int value = scanner.nextInt();
            tree.insert(value);
        }

        System.out.println(tree.diameter(tree.root));

        scanner.close();
    }
}
```

Question 32: Coin Change Problem

Problem Statement : Given a set of coins with certain denominations and a target amount, determine the number of ways to make the target amount using any combination of coins.

Input Format

The input begins with an integer N denoting the number of denominations. This is followed by N integers representing the denominations of coins. Then an integer target denoting the target amount.

Output Format

Print an integer representing the number of ways to make the target amount using the given coins.

Constraints

$1 \leq N \leq 100$

$1 \leq$ Denomination values ≤ 1000

$1 \leq$ Target amount ≤ 10000

Example

Sample Input 1

4
1 2 5 10
12

Sample Output 1

15

Sample Input 2

6
1 2 5 10 2 7
12

Sample Output 2

51

Explanation

For the given sample input 1, the denominations are 1, 2, 5, 10, and the target amount is 12. There are 15 different ways to make the amount 12 using these denominations.

Default Code

```
import java.util.*;

class Main {

   static int coinChange(int[] coins, int target) {

   }

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

```
        int n = scanner.nextInt();

        int[] coins = new int[n];

        for (int i = 0; i < n; i++) {
            coins[i] = scanner.nextInt();
        }

        int target = scanner.nextInt();

        System.out.println(coinChange(coins, target));

        scanner.close();
    }
}
```

Question 33: Top View of BST

Problem Statement : Given a Binary Search Tree (BST), find and print the top view(Left to right) of the BST.

Input Format
The input begins with an integer N denoting the number of nodes in the BST. This is followed by N pairs of integers representing the values and their respective depths of the nodes in the BST.

Output Format
Print the values of the nodes in the top view of the BST in a space-separated manner.

Constraints
$1 \leq N \leq 1000$
$-10^3 \leq$ Node values $\leq 10^3$

Example
Sample Input 1
7
50 0
30 1
70 1
20 2

40 2
60 2
80 2

Sample Output 1
20 30 50 70 80

Sample Input 2
3
50 0
30 1
70 1
Sample Output 2
30 50 70

Default Code

```
import java.util.*;

class Node {
   int data;
   int depth;
   Node left, right;


   Node(int value, int d) {
      data = value;
      depth = d;
      left = right = null;
   }
}

class Main {
   Node root;


   Main() {
      root = null;
   }
```

```
void insert(int value, int depth) {
   root = insertRec(root, value, depth);
}


Node insertRec(Node root, int value, int depth) {
   if (root == null)
      return new Node(value, depth);

   if (value < root.data)
      root.left = insertRec(root.left, value, depth + 1);
   else if (value > root.data)
      root.right = insertRec(root.right, value, depth + 1);

   return root;
}


void topView() {
   // Enter your code here
}


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


   int n = scanner.nextInt();

   Main tree = new Main();
   for (int i = 0; i < n; i++) {
      int value = scanner.nextInt();
      int depth = scanner.nextInt();
      tree.insert(value, depth);
   }


   tree.topView();

   scanner.close();
```

```
    }
}
```

Question 34: Egg Dropping Puzzle

Problem Statement : Given a certain number of floors and eggs, find the minimum number of attempts needed to determine the critical floor from which an egg breaks when dropped. You are allowed to break at most k eggs.

Input Format
The input contains two integers separated by space: N denoting the number of floors and K denoting the number of eggs.

Output Format
Print an integer representing the minimum number of attempts required to find the critical floor.

Constraints : $1 \leq N, K \leq 100$

Example
Sample Input 1
6 2

Sample Output 1
3

Sample Input 2
5 5

Sample Output 2
3

Explanation:
For the sample input 1, with 2 eggs and 6 floors, the minimum number of attempts needed to find the critical floor is 3. One approach to solve this is by dropping the eggs from the 3rd floor, then the 2nd floor if it doesn't break, and finally the 1st floor if needed.

Default Code

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

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

Question 35: Counting Bits

Problem Statement : Given an integer n, return *an array* ans *of length* n + 1 *such that for each* i (0 <= i <= n), ans[i] *is the* ***number of*** 1***'s*** *in the binary representation of* i.

Input Format
Integer: Only Integers

Output Format
Print the required answer

Constraints

0 <= n <= 10^5

Example

Sample Input 1
2

Sample Output 1
[0,1,1]

Sample Input 2
5

Sample Output 2
[0,1,1,2,1,2]

Question 36: Climbing Stairs

Problem Statement : You are climbing a staircase. It takes n steps to reach the top.Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

Input Format
Integer: Only Integers

Output Format
Print the required answer

Constraints

1 <= n <= 45

Example

Sample Input 1
5

Sample Output 1
8

Sample Input 2
13

Sample Output 2
377

Question 37: Character with maximum frequency in a String

Problem Statement

Input a String S. Write a function that returns the character with maximum frequency. Print the character returned. In case, if no character is repeated in the input, the function must return first character of the string. If the different characters have same frequency in the input, first character encountered must be returned.

Input Format

String: The Input String S must be lowercase alphabets only.

DO NOT INPUT Uppercase alphabets, numbers or alphanumeric characters.

Output Format

Character: For each test case, print the character with maximum frequency in a String.

Constraints

A string of length between 1 to 1000.

Example

Sample Input 1

abbc

Sample Output 1

b

Sample Input 2

aabbcc

Sample Output 2

a

Sample test case Explanation

In the first test case, b is the character with maximum frequency 2. b is returned and printed as output.

In the second test case a, b, c all three are of same frequency i.e 2 so first character a is encountered, returned and printed.

Question 38: Pattern Matching

Problem Statement

Input a pattern P and String S. Program must return “True” if S (String) follows the same pattern P, otherwise return “False”. Here follows means a full match, such that there is a one to one correspondence between a character/letter in pattern and a non-empty word in the String S.

Input Format

Pattern: String, can be a combination of alphabets or numbers.

String: String, a non-empty word that can be a combination of alphabets or numbers. Include space among each non-empty word.

First input Pattern and then String in the next line and they must not be same.

Output Format

Character: For each test case, return true or false.

Constraints

A string of length between 1 to 1000.

Example

Sample Input 1

abba

dog cat cat dog

Sample Output 1

true

Sample Input 2

abba

dog cat cat fish

Sample Output 2

false

Sample test case Explanation

In the first test case, dog is following a, cat is following b, so there is a full match. Output “true”

In the second test case dog is following a, cat is following b but fish is a mismatch. Not a full match, so output “false”

Question 39: Finding the Kth Largest Element in Array

Problem Statement

Given an integer array nums and an integer k, the task is to find and return the kth largest element in the array. It's essential to note that the element in question is the kth largest one in the sorted order, not necessarily the kth distinct element. The challenge is to solve this problem without utilizing sorting algorithms.

Input Format

Length/Size of array : Integer

Elements of array: Integer

Value of K: Integer

Output Format

Kth Largest Element in Array : Integer

Constraints

Do not use any sorting technique.

Example

Sample Input 1

6

3 2 1 5 6 4

2

Sample Output 1

5

Sample Input 2

9

3 2 3 1 2 4 5 5 6

4

Sample Output 2

4

Sample test case Explanation

In the first test case, 6 represent array size, 3 2 1 5 6 4 are array elements, then input K that is 2 to find the kth largest element in the array.

In the first test case, 9 represent array size, 3 2 3 1 2 4 5 5 6 are array elements, then input K that is 4 to find the kth largest element in the array.

Question 40: Prime Factors

Problem Statement: Input a number N. Find its prime factors in increasing order.

Input Format

A single Integer value ‘N’ whose prime factors are to be found.

Output Format

Print all prime factors in increasing order, The prime factors will all be integers.

Constraints

1 <= N <= 1000

Example

Sample Input 1

100

Sample Output 1

2 2 5 5

Sample Input 2

10

Sample Output 2

2 5

Sample Test Case Explanation

In the first test case, 100 represents the number for which prime factors are to be found.

Output is 2 x 2 x 5 x 5 = 100, making them the prime factors of 100.

In the second test case, 10 represents the number for which prime factors are to be found.

Output is 2 x 5 = 10, making them the prime factors of 10.

Question 41: Selection Sort

Problem Statement: Given an unsorted array of size N, use selection sort to sort arr[] in increasing order. You are also required to print the array after every pass (After every swap) using Arrays.toString().

Input Format

First Integer input (N), represents the size of the array.

Next line consist of N-space separated integers representing the values in the array

Output Format

(N-1) lines of array representation after each swap. The arrays must be printed using Arrays.toString().

Constraints

$1 \leq N \leq 10^3$

Example

Sample Input 1

5

12 23 54 67 17

Sample Output 1

[12, 23, 54, 67, 17]

[12, 17, 54, 67, 23]

[12, 17, 23, 67, 54]

[12, 17, 23, 54, 67]

Sample Input 2

4

45 12 90 65

Sample Output 2

[12, 45, 90, 65]

[12, 45, 90, 65]

[12, 45, 65, 90]

Question 42: Heapify

Problem Statement: Given an Array ‘arr’ of size ‘N’, implement the Heapify Algorithm so as to convert the sequence of elements into a Min Heap.

Input Format

First Line consists of Integer value (N) representing the size of Array.

Second line consists of N space-separated integers representing the values of the array.

Output Format

Print N space-separated integers representing the values of the Min Heap.

Constraints

$1 \leq N \leq 106$

$1 \leq arr[i] \leq 106$

Example

Sample Input 1

5

12 534 32 2 123

Sample Output 1

2 12 32 534 123

Sample Input 2

6

34 23 89 100 5 10

Sample Output 2

5 23 10 100 34 89

Question 43: Sort array indices

Problem Statement: Input an array of integers of size N. The task is to print the index of the largest number first and then the index of the 2nd largest number and so on till the last one. If two or more numbers are the same then print the index of the number which comes first in the array.

Input Format

First Line consists of Integer value (N) representing the size of Array.

Second line consists of N space-separated integers representing the values of the array.

Output Format

Print space separated Index of array elements as mentioned in question.

Sample Input 1

6

2 6 4 8 2 6

Sample Output 1

3 1 5 2 0 4

Sample Input 2

4

1 3 7 5

Sample Output 2

2 3 1 0

Question 44: Isomorphic Keyboard Layout Checker

Problem Statement: A company is designing a new keyboard layout, and they want to ensure that the layout is isomorphic. In this context, a keyboard layout is considered isomorphic if, when you press a key on the keyboard, it produces the same result regardless of whether the keyboard layout is QWERTY or AZERTY. The company has developed a program that checks whether two given strings represent isomorphic keyboard layouts.

Input Format

String: Two-line separated inputs will be given in String format.

Output Format

Check if layout is isomorphic or not, if isomorphic print “YES”, if not print “NO”.

Constraints

Do not Input special characters.

Do not start the input string with a number.

Sample Input 1

egg

add

Sample Output 1

YES

Sample Input 2

egg12

add11

Sample Output 2

NO

Question 45: Find a single element appearing once in a sorted array

Problem Statement: Input a sorted array consisting of only integers where every element appears exactly twice, except for one element which appears exactly once. Print the single element that appears only once.

Input Format

Input space-separated integers representing the values of the sorted array. Repeat only a single element in the sorted array.

Output Format

Print the element appearing once in the sorted array.

Constraints

Your solution must run in O(log n) time and O(1) space.

1 <= array.length <= 10^5

0 <= array[i] <= 10^5

Sample Input 1

1 1 2 2 3 4 4

Sample Output 1

3

Sample Input 2

1 1 2 3 3 4 4 8 8

Sample Output 2

2

Question 46: Starting Index of the Pattern matched in the String

Imagine you are developing a text processing application in Java. Users can input a large text document, and your application needs to perform pattern searching to find occurrences of specific words or phrases within the text. Write a function that returns the starting index of the pattern matched in the string first time.

Input Format

Line1: Text in which you want to search for a pattern. Each word must be separated by space in the text.

Line2: Pattern you want to search in text. Take care of case sensitivity during input of pattern.

Output Format

Print Index where pattern found, if no pattern found print -1.

Constraints

Implement KMP algorithm in code, complexity should not exceed O(m+n).

Sample Input 1

This is a simple example.

simple

Sample Output 1

10

Sample Input 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Ipsum

Sample Output 2

6