**ADI Coding Questions**

1. **Sorting Problem.**

Amrit have a list of students with their ID, First name and CGPA. Now he wants to rearrange them according to their CGPA in decreasing order. If two students have the same CGPA, then arrange them according to their first name in alphabetical order. If those two students also have the same first name, then order them according to their ID. No two students have the same ID. Help Amrit to generate the code for such kind of problem statement.

**Input Format**

* First line of input contains an integer, representing the total number of students.
* Second  line contains a list of student information in the following structure:
* ID Name CGPA
* Third line contains the output

**Constraints**

The name contains only lowercase English letters. It contains only integer numbers without leading zeros. The CGPA will contain, at most, 2 digits after the decimal point.

**Output Format**

After rearranging the students according to the above rules, print the first name of each student on a separate line.

**Sample Input**

5

33 Rumpa 3.68

85 Ashis 3.85

56 Samiha 3.75

19 Samara 3.75

22 Fahim 3.76

**Sample Output**

Ashis

Fahim

Samara

Samiha

Rumpa

**// Code**

import java.util.\*;

class Student{

    private int id;

    private String fname;

    private double cgpa;

    public Student(int id, String fname, double cgpa) {

        super();

        this.id = id;

        this.fname = fname;

        this.cgpa = cgpa;

    }

    public int getId() {

        return id;

    }

    public String getFname() {

        return fname;

    }

    public double getCgpa() {

        return cgpa;

    }

}

//Complete the code

public class Solution

{

    public static void main(String[] args){

        Scanner in = new Scanner(System.in);

        int testCases = in.nextInt();

        List<Student> studentList = new ArrayList<Student>();

        while(testCases>0){

            int id = in.nextInt();

            String fname = in.next();

            double cgpa = in.nextDouble();

            Student st = new Student(id, fname, cgpa);

            studentList.add(st);

            testCases--;

        }

      Collections.sort(studentList,new Comparator<Student>(){

        public int compare(Student a,Student b){

        if(a.getCgpa()!=b.getCgpa()){

            return Double.compare(b.getCgpa(),a.getCgpa());

        }else{

            if(!a.getFname().equals(b.getFname())){

                return a.getFname().compareTo(b.getFname());

            }else{

                return a.getId()-b.getId();

            }

        }

    }

      });

          for(Student st: studentList){

            System.out.println(st.getFname());

        }

    }

}

1. **Array : SubArray**

We define the following:

* A subarray of an -element array is an array composed from a contiguous block of the original array's elements. For example, if , then the subarrays are , , , , , and . Something like  would not be a subarray as it's not a contiguous subsection of the original array.
* The sum of an array is the total sum of its elements.
  + An array's sum is negative if the total sum of its elements is negative.
  + An array's sum is positive if the total sum of its elements is positive.

Given an array of  integers, find and print its number of negative subarrays on a new line.

**Input Format**

The first line contains a single integer, , denoting the length of array .  
The second line contains  space-separated integers describing each respective element, , in array .

**Constraints**

**Output Format**

Print the number of subarrays of  having negative sums.

**Sample Input**

5

1 -2 4 -5 1

**Sample Output**

9

**// Code**

import java.io.\*;

import java.util.\*;

public class Solution

{

    public static void main(String[] args) {

       Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        int sumOfNeg = 0;

        int[] arr = new int[n];

        for (int i=0; i<n; i++) {

            int num = scan.nextInt();

            // get negative values for array size 1

            if (num < 0) sumOfNeg++;

            arr[i] = num;

        }

        scan.close();

        // start getting negative sum values for array size 2

        int subArraySize = 2;

        for (int i=1; i<n; i++) {

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

                int[] subArr = Arrays.copyOfRange(arr, j, j+subArraySize);

                if (Arrays.stream(subArr).sum() < 0) sumOfNeg++;

            }

            subArraySize++;

        }

        System.out.println(sumOfNeg);

    }

}

1. **String Anagrams:**

Amrit is taking PA class and during taking attendance he observes that many students have different name but same characters in different order. So he decides to print all the students who have similar no of characters but different names. When two names/strings have same characters but in different order, it is called as anagram strings.

Two strings,  Str1 and  Str2, are called anagrams if they contain all the same characters in the same frequencies. For this challenge, the test is not case-sensitive. For example, the anagrams of CAT are CAT, ACT, tac, TCA, aTC, and CtA.

**Input Format**

The first line contains a string St1.  
The second line contains a string Str2.

**Constraints**

* Strings  Str1 and  Str2 consist of English alphabetic characters.
* The comparison should NOT be case sensitive.

**Sample Input 1**

anagram

margana

**Sample Output 1**

Anagrams

**Explanation 1**

| **Character** | **Frequency:**anagram | **Frequency:**margana |
| --- | --- | --- |
| A or a | 3 | 3 |
| G or g | 1 | 1 |
| N or n | 1 | 1 |
| M or m | 1 | 1 |
| R or r | 1 | 1 |

The two strings contain all the same letters in the same frequencies, so we print "Anagrams".

**Sample Input 1**

anagramm

marganaa

**Sample Output 1**

Not Anagrams

**Explanation 1**

| **Character** | **Frequency:**anagram | **Frequency:**marganaa |
| --- | --- | --- |
| A or a | 3 | 4 |
| G or g | 1 | 1 |
| N or n | 1 | 1 |
| M or m | 2 | 1 |
| R or r | 1 | 1 |

The two strings don't contain the same number of a's and m's, so we print "Not Anagrams".

**Sample Input 2**

Hello

hello

**Sample Output 2**

Anagrams

**// Code**

import java.util.Scanner;

public class Solution {

    static boolean isAnagram(String a, String b)

    {

        boolean b1=false;

    int x = a.length();

        int y = b.length();

        if(x==y){

            char[] A1 = a.toLowerCase().toCharArray();

            char[] B1 = b.toLowerCase().toCharArray();

            java.util.Arrays.sort(A1);

            java.util.Arrays.sort(B1);

            if(java.util.Arrays.equals(A1, B1))

              b1=true;

        }

        return b1;

    }

  public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        String a = scan.next();

        String b = scan.next();

        scan.close();

        boolean ret = isAnagram(a, b);

        System.out.println( (ret) ? "Anagrams" : "Not Anagrams" );

    }

}

1. **Newborn’s Name**

A couple had a baby 3 days ago and they have been thinking over what to name their baby so they came up with an idea that they should merge their first names and come with a name. But, they are afraid that it will be a long name. Help them figure out how small they can come up with a name if they merge their names such that they get both their names as subsequences in the child’s name.

**Sample Input 1:**

nisha antil

**Sample Output 1:**

7

**Sample Input 2:**

maahi ahima

**Sample Output 2:**

7

**Input Explanation:**

Space separated first name of couple

**Output Explanation:**

Number of characters in baby’s name

In Sample 1, shortest name could be “antilsha” of length “8” which has “antil” and “nisha” as subsequence.

In Sample 2, shortest name could be “maahima”of length “7” which has “mahi” and “ahima” as subsequence..

**// Code : (remove duplicate characters and print total length of two strings)**

**import** java.util.HashSet;

**import** java.util.Set;

**public** **class** RemoveDuplicasyInString

{

**public** **static** **void** main(String[] args)

{

String s1="antil";

String s2="nisha";

**int** output = *removeDuplicasy*(s1,s2);

System.***out***.println("Output :"+output);

}

**public** **static** **int** removeDuplicasy(String s1,String s2)

{

String s3=s1+s2;

Set s = **new** HashSet();

**for**(**int** i=0;i<s3.length();i++)

{

s.add(s3.charAt(i));

}

**return** s.size();

}

}

1. **Binary Search Problem.**

In a classroom there are N no of students, every student have a roll no and all students sit in their classroom according to their roll nos into ascending order. Now parents of a student (name : Anil) want to meet their son and goto the the concerned class room and start finding out their son.

Design an algorithm to help the parents of students anil to check, whether anil is present into class or not.

**Program take 2 inputs:**

First, n , no of students.

Second, list of roll nos of all the students presnt into the classroom.

Third, n1, roll no of the target student, parents are searching.

**Sample Input 1**

**10**

**2 3 1 5 7 8 9 12 4 15**

**5**

**Sample Output 1**

**“Present”**

**Sample Input 2**

**5**

**6 7 8 9 12**

**14**

**Sample Output 1**

**“Not Present”**

**Note: n >0 and n1>0 , Parents already know the roll no of their son.**

**Code:**

// Binary Search Algorithm.

**import** java.util.\*;

**public** **class** BinarySearch

{

**public** **static** **void** main(String[] args)

{

**int** n,n1,i;

Scanner s = **new** Scanner(System.***in***);

System.***out***.println("Enter the no of stduents :");

n=s.nextInt();

**int** st[] = **new** **int**[n];

System.***out***.println("Enter the roll nos of students, present into class :");

**for**(i=0;i<n;i++)

{

st[i]=s.nextInt();

}

System.***out***.println("Enter the roll no, parents want to search :");

n1 = s.nextInt();

Arrays.*sort*(st);

/\*\*\*\*\*\*\*\*\*\*\* Logic \*\*\*\*\*\*\*\*\*\*/

**int** found=0;

**int** first=0;

**int** last=st.length-1;

**int** middle=(first+last)/2;

**while**(first<=last)

{

**if**(n1<st[middle])

{

last=middle-1;

middle=(first+last)/2;

}

**if**(n1>st[middle])

{

first=middle+1;

middle=(first+last)/2;

}

**if**(n1==st[middle])

{

found=1;

**break**;

}

System.***out***.println("First : "+first);

System.***out***.println("last :" +last);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**if**(found==1)

System.***out***.println("Present");

**else**

System.***out***.println("Not Present");

}

}

1. **Binary Search Problem-2**

Amrit have a bucket with some boxes with numbers, characters and fractional values over the boxes. Now Amrit want to search specific box of number, character and fraction value.

Make an algorithm to implement the searching ( Binary Searching) by taking input of int, char and fraction value from the user and check whether they are present or not. If they are present, print also their index value.

**Take three input from user :**

* Integer n,
* Character ch,
* Double d
* And check whether this data is present into the given record of int, char and double arrays by using Binary Searching Approach.

**Note :** We can use Arrays.binarySearch() inbuilt method to implement it.

**Sample Input 1**

**===== Given data =====**

Int array : 1 2 2 3 4 5 7 8 9 11 31 34 89

Char array : d e p t y

Double array : 1.2 2.3 4.5 5.6 6.2

Enter the int no you want to search :

5

Enter the character you want to search :

e

Enter the double no, you want to search :

2.3

**Sample Output 1**

5 found at index :5

e found at index :1

2.3 found at index :1

==================================================

**Sample Input 2**

===== Given data =====

Int array : 1 2 2 3 4 5 7 8 9 11 31 34 89

Char array : d e p t y

Double array : 1.2 2.3 4.5 5.6 6.2

Enter the int no you want to search :

7

Enter the character you want to search :

d

Enter the float no, you want to search :

4.5

Sample Output 2

7 found at index :6

d found at index :0

4.5 found at index :2

//Code

**import** java.util.Arrays;

**import** java.util.Scanner;

**import** java.io.\*;

// Binary Search : by using Arrays.binarySearch() method

**public** **class** BinarySearchIterative

{

**public** **static** **void** main(String[] args) **throws** Exception

{

**int** a[]= {34,89,5,4,31,2,11,1,2,3,7,8,9};

**char** ch[]= {'t','d','y','p','e'};

**double** d[]= {1.2,2.3,4.5,5.6,6.2};

Arrays.*sort*(a);

Arrays.*sort*(ch);

Arrays.*sort*(d);

System.***out***.println("===== Given data =====");

System.***out***.print("Int array :");

**for**(**int** i:a)

System.***out***.print(" "+i);

System.***out***.print("\nChar array :");

**for**(**char** i:ch)

System.***out***.print(" "+i);

System.***out***.print("\nDouble array :");

**for**(**double** i:d)

System.***out***.print(" "+i);

Scanner s = **new** Scanner(System.***in***);

System.***out***.println("\nEnter the int no you want to search :");

**int** n = s.nextInt();

System.***out***.println("\nEnter the character you want to search :");

**char** ch1 = (**char**)System.***in***.read();

System.***out***.println("\nEnter the double no, you want to search :");

**double** f = s.nextDouble();

System.***out***.println(n+" found at index :"+Arrays.*binarySearch*(a,n));

System.***out***.println(ch1+" found at index :"+Arrays.*binarySearch*(ch,ch1));

System.***out***.println(f+" found at index :"+Arrays.*binarySearch*(d,f));

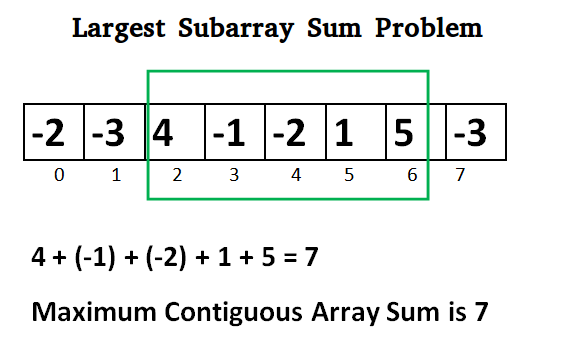
}

}

1. Amrit have 2 buckets, in bucket-1 there are some boxes with positive and negative numbers, and bucket-2 is empty. Now Amrit want to pic some boxes and put into bucket-2 but in one condition, that when Amrit add all numbers put in bucket-2, sum will be largest or we can say value of bucket-2 will be largest. We can solve this problem by using Kadane’s Algorithm, to make largest value subarray.

For Example:

Given an array **arr[]** of size **N**.The task is to find the sum of the contiguous subarray within a **arr[]** with the largest sum.



Sample Input 1

8

-2 -3 4 -1 -2 1 5 -3

Sample Output 1

7

//Code

// Kadane's Algorithm.

// Max sub array.

**import** java.util.\*;

**public** **class** MaxSubArray

{

**public** **static** **void** main(String[] args)

{

Scanner s = **new** Scanner(System.***in***);

System.***out***.println("\nEnter the no of elements :");

**int** n=s.nextInt();

**int** a[]=**new** **int**[n];

System.***out***.println("\nEnter the array elements :");

**for**(**int** i=0;i<n;i++)

{

a[i]=s.nextInt();

}

**int** maxSum = *maxSubArray*(a);

System.***out***.println("Max sub Array :"+maxSum);

}

**public** **static** **int** maxSubArray(**int** a[])

{

**int** sum=0,target=0;

**for**(**int** i=0;i<a.length;i++)

{

sum+=a[i];

**if**(sum>target)

target=sum;

**if**(sum<0)

sum=0;

}

**return** target;

}

}

1. Amrit is standing on the ticket winodw of a **Movie Ticket Counter,** and looking that people are standing in a queue, but this is **Double ended queue** means peoples can be added either into the last position or into the first position. So design an algorithm for this concept.

Note : We can use java.util.Deque inbuilt interface for this purpose.

Sample Input and Output:

Adding 1,2:[1, 2]

Removing 1,2

Adding 3,4 :[3, 4]

**// Code**

**import** java.util.\*;

**public** **class** DequeDemo

{

**public** **static** **void** main(String[] args)

{

Deque <Integer>d = **new** ArrayDeque();

d.addFirst(1);

d.addLast(2);

System.***out***.println("\nAdding 1,2:"+d);

**int** f = d.removeFirst();

**int** l = d.removeLast();

System.***out***.println("\nRemoving 1,2");

d.add(3);

d.add(4);

System.***out***.println("\nAdding 3,4 :"+d);

}

}

1. **Strings**

**Amrit** have a bucket, and bucket contains various N no of boxes and some names (strings) are written on that boxes, Now he want to get only that boxes which contain the palindrome string. Now write a logic and help Mr Amrit to tell, how many boxes he should get from bucket.

Input N contain no of strings, and then give N no of strings and write the logic to find the palindrome strings only, and print the no of boxes, amrit will get from the bucket.

**Sample Input:**

5

abc abca abccba abcbc bcaabc

**Sample Output:**

2

1. **String**

Amrit want to print all those prime numbers from n1 to n2, which have even sum when we add individual digits of that prime no like: 15=1+5=6 (even)

Note: 73 is prime, but digits sum is odd, 7+3=10 again 1+0=1 (odd).

Input **n1(start)** and **n2(end)**, and print all prime nos which contains even sum of all the individual digits.

**Sample Output:**

1

20

**Sample Output:**

11 13 17

**(GCD)**

1. **Title:** **Greatest Common Divisor of two numbers**

**Difficulty Level: Medium**

Amrit is making the list of students roll no wise. He wants to calculate the greatest common divisor among the smallest and greatest roll nos. from list of given students. Help Amrit to generate the code and print the required output.

Given an integer array nums, return the greatest common divisor of the smallest number and largest number in nums.

The greatest common divisor of two numbers is the largest positive integer that evenly divides both numbers.

**Example:**

**Input:** nums = [2,5,6,9,10]

**Output:**2

**Explanation:**

The smallest number in nums is 2.

The largest number in nums is 10.

The greatest common divisor of 2 and 10 is 2.

**First Line**: contains the N.

**Second Line**: contains the N elements

**Third Line:** contains the output

**Constraints:**

2 <= nums.length <= 1000

1 <= nums[i] <= 1000

**Sample Input1:**

5

2 5 6 9 10

**Sample Output1:**

2

**Sample Input2:**

5

2 7 8 9 12

**Sample Output2:**

2

**Sollution:**

import java.util.\*;

class GCDOfMinMax

{

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);

int n=s.nextInt();

int nums[] = new int[n];

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

nums[i]=s.nextInt();

int result = findGCDOfMinMax(nums);

System.out.println(result);

}

public static int findGCDOfMinMax(int[] nums) {

int min = Arrays.stream(nums).min().orElse(Integer.MAX\_VALUE);

int max = Arrays.stream(nums).max().orElse(Integer.MIN\_VALUE);

return gcd(min, max);

}

// Euclidean algorithm to find GCD

public static int gcd(int a, int b) {

if (b == 0) {

return a;

}

return gcd(b, a % b);

}

}

**(String with Recursion)**

1. **Title: Permutations of the given String.**

**Difficulty Level: Medium**

Soft skills teacher want to discuss all possible combinations of a word with students. He gave a word to students and asking about the solution. Help students to make the code, which input a word and make all the combinations of that word.

**Explanation**:

Input a string like “ab” and output should be “ab ba” .

**First Line** contain the input (String)

**Second Line** contain the output (String)

**Sample Input1:**

ab

**Sample Output 1:**

ab

ba

**Sample Input2:**

123

**Sample Output 2:**

123

132

231

213

321

312

**Solution:**

import java.util.Scanner;

class BackTracking

{

public static void main(String args[])

{

Scanner s = new Scanner(System.in);

String input=s.next();

printPermutation(input,"");

}

public static void printPermutation(String str,String output)

{

if(str.length()==0)

{

System.out.println(output);

return;

}

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

{

char cc=str.charAt(i);

String ns=str.substring(0,i)+str.substring(i+1);

printPermutation(ns,output+cc);

}

}

}

**(String Pattern matching algorithm)**

1. **Title: Check whether the given pattern is present into the given string or not.**

**Difficulty Level: Medium**

English teacher is teaching a lesson to students. He found many characters/words are in repeated manner. So he ask to all students, to check whether the given pattern/characters are present into another sentence or not. So help students to generate the code which take a sentence “S” and searching pattern “P” and check whether the given pattern is present into the sentence or not.

**First Line** contains the input String.

**Second Line** contains the searching pattern/characters/string

**Third Line** contains the output.

**Note:** Input Output String is case sensitive. Case must be matched.

**Sample Input 1:**

abc abcabcabc

abc

**Sample Output 1:**

present

**Sample Input 2:**

Hello Students

students

**Sample Output 2:**

not present

**Solution:**

import java.util.\*;

class PatternChecker {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

String inputString = s.nextLine();

String pattern = s.nextLine();

boolean containsPattern = checkPattern(inputString, pattern);

if (containsPattern) {

System.out.println("present");

} else {

System.out.println("not present");

}

}

public static boolean checkPattern(String inputString, String pattern) {

return inputString.contains(pattern);

}

}

1. **Title: Print the smallest positive Integer.**

**Difficulty Level: Medium**

Amrit is teaching into class and given a list of numbers to all students and ask about the smallest positive integer by using that list of numbers. Help students to create the code which takes a list of numbers from user and print the smallest positive integer with the help of that list.

**First Line** contains the N

**Second Line** contains the N numbers

**Third Line** contains the Output

**Sample Input 1:**

5

1 2 3 4 0

**Sample Output 1**

5

**Sample Input 2**

3

2 3 0

**Sample Output 2**

1

**Solution:**

import java.util.\*;

class SmallestMissingPositive {

public static int firstMissingPositive(int[] nums) {

int n = nums.length;

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

while (nums[i] > 0 && nums[i] <= n && nums[nums[i] - 1] != nums[i]) {

// Swap elements to put them in their correct positions

int temp = nums[nums[i] - 1];

nums[nums[i] - 1] = nums[i];

nums[i] = temp;

}

}

// Find the first index where nums[index] != index + 1

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

if (nums[i] != i + 1) {

return i + 1; // Return the missing positive integer

}

}

return n + 1; // If all positive integers from 1 to n are present, return n + 1

}

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

int n=s.nextInt();

int[] nums = new int[n];

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

nums[i]=s.nextInt();

int result = firstMissingPositive(nums);

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

}

}

1. **Title: Longest common subsequence.**

**Difficulty Level: Advance**

English teacher given text1 and text2 to students and asked about the longest common subsequence among given text1 and text2. Now help students to generate the code for such kind of problem statement.

Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common subsequence, return 0.

A subsequence of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.

**For example**, "ace" is a subsequence of "abcde".

A common subsequence of two strings is a subsequence that is common to both strings.

**Example:**

**Input:** text1 = "abcde", text2 = "ace"

**Output:** 3

**Explanation:** The longest common subsequence is "ace" and its length is 3.

**Constraints:**

1 <= text1.length, text2.length <= 1000

text1 and text2 consist of only lowercase English characters.

**First Line** contains the first text/string.

**Second Line** contains the second text/string

**Third Line** contains the output.

**Sample Input1:**

Abc

abc

**Sample Output1:**

2

**Sample Input2**:

Abc de

cd

**Sample Output1:**

2

**Solution:**

// Longest Common Subsequence among two given strings.

import java.util.\*;

class LongestCommonSubsequence {

public static int longestCommonSubsequence(String text1, String text2) {

int m = text1.length();

int n = text2.length();

int[][] dp = new int[m + 1][n + 1];

for (int i = 0; i <= m; i++) {

for (int j = 0; j <= n; j++) {

if (i == 0 || j == 0) {

dp[i][j] = 0; // Initializing the first row and column to 0

} else if (text1.charAt(i - 1) == text2.charAt(j - 1)) {

dp[i][j] = dp[i - 1][j - 1] + 1; // If characters match, increment the count

} else {

dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]); // If characters don't match, take max from top or left

}

}

}

return dp[m][n]; // Return the length of longest common subsequence

}

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

String text1 = s.nextLine();

String text2 = s.nextLine();

int result = longestCommonSubsequence(text1, text2);

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

}

}

**(Binary Tree)**

1. **Title: To check whether the given binary tree is symmetric or not.**

**Difficulty Level: Advance**

Amrit is teaching about binary tree into class. He draw a tree structure on board and give some data (elements) to tree, as shown in figure below:

**1**

**/ \**

**2 2**

**/ \ / \**

**3 4 4 3**

Now he found that, given tree is symmetric. Symmetry in this case means that the left sub-tree is a mirror reflection of the right sub-tree.

Now he gave an input contains integer data elements and ask students that, it is symmetric or not. Output should be either "true" (if symmetric) or "false" (if not).

**Example:**

**Input :** 1223443

**Output:** true

**Note:** Input Length must be greater than 0 and less than equal to 7.

0<Input Length<=7

**Sample Input1:**

2445665

**Sample Output1:**

true

**Sample Input2:**

2445667

**Sample Output2:**

false

**Solution:**

// Symmetric Binary Tree.

import java.util.Scanner;

class TreeNode {

int val;

TreeNode left;

TreeNode right;

TreeNode(int x) {

val = x;

}

}

class SymmetricBinaryTree {

public boolean isSymmetric(TreeNode root) {

if (root == null) {

return true;

}

return isSymmetricHelper(root.left, root.right);

}

private boolean isSymmetricHelper(TreeNode left, TreeNode right) {

if (left == null && right == null) {

return true;

}

if (left == null || right == null) {

return false;

}

if (left.val != right.val) {

return false;

}

return isSymmetricHelper(left.left, right.right) && isSymmetricHelper(left.right, right.left);

}

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

String str=s.next(); //str=1223443

TreeNode root = new TreeNode(Integer.parseInt(""+str.charAt(0)));

root.left = new TreeNode(Integer.parseInt(""+str.charAt(1)));

root.right = new TreeNode(Integer.parseInt(""+str.charAt(2)));

root.left.left = new TreeNode(Integer.parseInt(""+str.charAt(3)));

root.left.right = new TreeNode(Integer.parseInt(""+str.charAt(4)));

root.right.left = new TreeNode(Integer.parseInt(""+str.charAt(5)));

root.right.right = new TreeNode(Integer.parseInt(""+str.charAt(6)));

SymmetricBinaryTree solution = new SymmetricBinaryTree();

boolean isSymmetric = solution.isSymmetric(root);

System.out.println(isSymmetric);

}

}

**(Dynamic Programming)**

1. **Title: Searching a pattern, into the given string with the help of “.” And “\*” characters.**

**Difficulty Level: Advance**

Amrit have many files into his PC. Sometimes he remembered the complete file name but some time not. Now he wants to search files. But he faces issues to search those files, which he remembered any single character. Now he want to create an algorithm through which he can know about file is present or not in PC, even by giving single character.

**For example**, any file present from name "first" and

When he search "first" pc give result "true",

And when he search "f", pc give result "false"

But when he search "f\*" pc give result "true",

And again he search ".\*" again pc can give result "true".

**"." mean any single character**

**"\*" mean any no of characters of same as previous type.**

Help Amrit to generate such kind of code.

Given an input string s and a pattern p, implement regular expression matching with support for '.' and '\*' where:

**'.' Matches any single character.​​​​**

**'\*' Matches zero or more character(s) same as of the preceding element.**

The matching should cover the entire input string (not partial).

**First Line,** contains the String "S"

**Second Line,** contains the searching pattern "P"

**Third Line,** contains the output

**Example 1:**

**Input:** s = "aa", p = "a"

**Output:** false

**Explanation:** "a" does not match the entire string "aa".

**Example 2:**

**Input:** s = "aaaaa", p = "a\*"

**Output:** true

**Explanation:** '\*' means zero or more of the preceding element, **'a'**. Therefore, by repeating **'a'** once, it becomes **"aa".**

**Example 3:**

**Input:** s = "ab", p = ".\*"

**Output:** true

**Explanation:** ".\*" means "zero or more (\*) of any character (.)".

**Example 4:**

**Input:** s = "Abc", p = "A.."

**Output:** true

**Explanation:** Here “..” mean any two characters.

**Constraints:**

* 1 <= s.length <= 20
* 1 <= p.length <= 20
* s contains only lowercase English letters.
* p contains only lowercase English letters, '.', and '\*'.
* It is guaranteed for each appearance of the character '\*', there will be a previous valid character to match.

**Sample Input1:**

aa

a\*

**Sample Output1:**

true

**Sample Input2:**

Abc

Ab\*

**Sample Output2:**

false

**Solution:**

import java.util.Scanner;

class FileSearch {

public static boolean isMatch(String s, String p) {

if (p.isEmpty()) {

return s.isEmpty();

}

boolean firstMatch = !s.isEmpty() && (s.charAt(0) == p.charAt(0) || p.charAt(0) == '.');

if (p.length() >= 2 && p.charAt(1) == '\*') {

return (isMatch(s, p.substring(2)) || (firstMatch && isMatch(s.substring(1), p)));

} else {

return firstMatch && isMatch(s.substring(1), p.substring(1));

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String s = scanner.nextLine();

String p = scanner.nextLine();

boolean output = isMatch(s, p);

System.out.println(output);

}

}