/\*Write a program to check if a given string is a valid palindrome, considering only alphanumeric characters and ignoring cases.

A palindrome is a word, phrase, number, or other sequence of characters that reads the same forward and backward.

Your program should return True if the input string is a valid palindrome, and False otherwise.

For example, if the input string is "A man, a plan, a canal: Panama", the program should return True, as the string is a

valid palindrome when considering only alphanumeric characters ("amanaplanacanalpanama").

Write a program that takes a string as input and determines whether it is a valid palindrome or not, considering alphanumeric

characters and ignoring cases.

Sample input: "A man, a plan, a canal: Panama"

Sample output: True

Sample input: "hello world"

Sample output: False

\*/

import java.util.\*;

public class Alphanumeric\_Palindrome {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s = sc.nextLine().toLowerCase();

char[] ch = new char[0];

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

if(Character.isLetter(s.charAt(i)))

{

ch = Arrays.copyOf(ch,ch.length+1);

ch[ch.length-1] = s.charAt(i);

}

}

//System.out.println(Arrays.toString(ch));

boolean check = true;

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

if(ch[i] != ch[ch.length-1-i])

{

check = false;

break;

}

}

if(check == true)

{

System.out.println("True");

}

else

{

System.out.println("False");

}

}

}

2. /\*Write a program to check if two strings are anagrams of each other. An anagram is a word or phrase formed by rearranging

the letters of another word or phrase. Your program should return True if the two input strings are anagrams, and False

otherwise.

For example, if the input strings are "listen" and "silent", the program should return True, as both strings can be rearranged

to form the same letters.

Write a program that takes two strings as input and determines whether they are anagrams or not.

Sample input1:

----------------

listen

silent

Sample output1:

---------------

True

Sample input2:

---------------

hello

world

Sample output2:

---------------

False

\*/

import java.util.\*;

public class Anagram {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.nextLine();

char[] ch1 = s1.toCharArray();

String s2 = sc.nextLine();

char[] ch2 = s2.toCharArray();

int check = 0;

for (int i = 0; i < ch1.length; i++) {

for (int j = 0; j < ch2.length; j++) {

if(ch1[i] == ch2[j])

{

check ++;

break;

}

}

}

if(ch1.length == ch2.length && check == ch1.length)

{

System.out.println("True");

}

else

{

System.out.println("False");

}

}

}

3. /\* Take last digit of an array and placed it at front.

\* The process run upto n times by user input

\*

\* Input :

\* ----------------

\* 5

\* 1 2 3 4 5

\* 3

\*

\* Output :

\* ----------------

\* 3 4 5 1 2

\*/

import java.util.\*;

public class Arrays\_Rotation {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int[] arr = new int[n];

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

{

arr[i] = sc.nextInt(); // [1,2,3,4,5]

}

int k = sc.nextInt(); // k = 2

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

{

int num = arr[n-1]; // num = 5,4

for (int j = n-1; j>0; j--) {

arr[j] = arr[j-1];

}

arr[0] = num;

}

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

System.out.print(arr[i]+" "); // [4,5,1,2,3]

}

}

}

4. /\* Count of Nonrepeat-digited number (10, 12, 13,.............) between a range

\* Note:

\* Repeat-Digit :- 11, 22, 33, .............

\* You should ignore the digit and count the other numbers

\*

\* Input :

\* ---------------------

10

25

Output :

-------------------

14

\*/

import java.util.\*;

public class BetweenNumber {

public static void main(String[] args) {

Scanner sc= new Scanner(System.in);

int a = sc.nextInt();

int b = sc.nextInt();

int count=0;

for (int i = a; i <=b; i++) {

String s = Integer.toString(i);

int num=0;

for (int j = 0; j < s.length()-1; j++)

{

for (int k = j+1; k < s.length(); k++)

{

if(s.charAt(j)==s.charAt(k))

{

num++;

}

}

}

if(num==0)

{

count++;

}

}

System.out.println(count);

}

}

5. /\*Write a program to find the longest common prefix among an array of strings. The longest common prefix is the string that

all strings in the array have in common as a prefix.

For example, given the array of strings ["flower", "flow", "flight"], the program should return the string "fl" as it is the

longest common prefix among all the strings.

Write a program that takes an array of strings as input and returns the longest common prefix among them. \*/

import java.util.\*;

public class Common\_Prefix {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Number of words: ");

int n = sc.nextInt();sc.nextLine();

String[] arr = new String[n];

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

{

arr[i] = sc.nextLine();

}

String ans = "";

int min = arr[0].length();

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

if(arr[i].length()<min)

{

min = arr[i].length();

}

}

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

boolean check = true;

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

if(arr[j-1].charAt(i) != arr[j].charAt(i))

{

check = false;

break;

}

}

if(check == true)

{

ans += arr[0].charAt(i);

}

else

{

break;

}

}

System.out.println(ans);

}

}