Q1.

One of the important aspect of object oriented programming is readability of the code. To enhance the readability of code, developers write function and variable names in Camel Case. You are given a string, S, written in Camel Case. FindAllTheWordsContainedInIt.

Input Format

A single line contains the string.

Constraints

|S|<=1000

Output Format

Print words present in the string, in the order in which it appears in the string.

Sample Input

```
IAmACompetitiveProgrammer
```

Sample Output

```
I
Am
A
Competitive
Programmer
```

Explanation

There are 5 words in the string.

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

        for(int i=0; i<length-1; i++){
            if(is_capital(s.charAt(i+1)) && i<length-1){
                System.out.println(s.charAt(i));
            }
            else{
                System.out.print(s.charAt(i));
            }
        }
        System.out.print(s.charAt(length-1));
    }

    public static boolean is_capital(char c){
        if( (int)(c) >= 'A' && (int)(c) <= 'Z'){
            return true;
        }
        return false;
    }
}
```

Deepak and Gautam are having a discussion on a new type of number that they call Coding Blocks Number or CB Number. They use following criteria to define a CB Number.

1. 0 and 1 are not a CB number.
2. 2,3,5,7,11,13,17,19,23,29 are CB numbers.
3. Any number not divisible by the numbers in point 2(Given above) are also CB numbers.

Deepak said he loved CB numbers.Hearing it, Gautam throws a challenge to him.

Gautam will give Deepak a string of digits. Deepak's task is to find the number of CB numbers in the string.

CB number once detected should not be sub-string or super-string of any other CB number.
Ex- In **4991**, both **499** and **991** are CB numbers but you can choose either **499** or **991**, not both.

Further, the CB number formed can only be a sub-string of the string.
Ex - In **481**, you can not take **41** as CB number because 41 is not a sub-string of **481**.

As there can be multiple solutions, Gautam asks Deepak to find the maximum number of CB numbers that can be formed from the given string.

Deepak has to take class of Launchpad students. Help him by solving Gautam's challenge.

Input Format

First line contain size of the string.

Next line is A string of digits.

Constraints

1 <= Length of strings of digits <= 17

Output Format

Maximum number of CB numbers that can be formed.

Sample Input

```
5
81615
```

Sample Output

```
2
```

Explanation

61 and 5 are two CB numbers.

```java
import java.util.*;
public class Main {
public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc= new Scanner (System.in);
        int l = sc.nextInt();
        String s = sc.next();
        Substring(s);
    }

    public static void Substring(String s) {
        int count=0;
        boolean [] visited = new boolean[s.length()];
        for(int len =1; len<=s.length(); len++) {
            for(int j=len; j<= s.length(); j++) {
                int i = j-len;

                String s1 = s.substring(i,j);
                long num = Long.parseLong(s1);

                if(is_cb_number(num) &&
is_visited(visited,i, j-1)) {
                    count++;

                    for(int k=i; k<=j-1; k++) {
                        visited[k]=true;
                    }
                }
            }

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

```
    public static boolean is_cb_number(long num) {
        if(num==0 || num==1) {
            return false;
        }
        int [] arr = {2,3,5,7,11,13,17,19,23,29};

        for(int i=0; i<arr.length; i++) {
            if(num == arr[i]) {
                return true;
            }
            else if(num % arr[i] == 0) {
                return false;
            }
        }


    return true;
    }

    public static boolean is_visited(boolean [] visited,
int i, int j) {
        for(int k=i; k<=j; k++) {
            if(visited[k] == true) {
                return false;
            }
        }
        return true;
    }

}
```

Take as input S, a string. Write a function that does basic string compression. Print the value returned. E.g. for input “aaabbccds” print out a3b2c2d1s1.

Input Format

A single String S

Constraints

1 < = length of String < = 1000

Output Format

The compressed String.

Sample Input

```
aaabbccds
```

Sample Output

```
a3b2c2d1s1
```

Explanation

In the given sample test case 'a' is repeated 3 times consecutively, 'b' is repeated twice, 'c' is repeated twice and 'd and 's' occurred only once.

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

        int length  = s.length();
        int i = 0;
        if(length == 1){
            System.out.print(s);
            System.out.print(1);
        }
        else{
            while(i<length){
                char c = s.charAt(i);
                int count=1;
                if(i == length-1 && c == s.charAt(i-1)){
                    count++;
                    break;
                }
                if(i<length-1){
                    while(c == s.charAt(i+1)){
                        count++;
                        if(i<length-2){
                            i++;
                        }
                        else{
                            break;
                        }
                    }
                }
                System.out.print(c);
                System.out.print(count);
                i++;
            }
        }
    }
}
```

Take as input S, a string. Write a function that toggles the case of all characters in the string. Print the value returned.

Input Format

String

Constraints

Length of string should be between 1 to 1000.

Output Format

String

Sample Input

```
abC
```

Sample Output

```
ABc
```

Explanation

Toggle Case means to change UpperCase character to LowerCase character and vice-versa.

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

    public static void toggle(String s){
        int length = s.length();

        for(int i = 0; i< length; i++){
            int i_value = (int)(s.charAt(i));
            if( i_value <= 'Z' && i_value >= 'A'){
                i_value+=32;
                System.out.print((char)i_value);
            }
            else{
                i_value-=32;
                System.out.print((char)i_value);
            }
        }
    }
}
```

Take as input S, a string. Write a function that does basic string compression. Print the value returned. E.g. for input “aaabbccds” print out a3b2c2ds.

Input Format

A single String S.

Constraints

A string of length between 1 to 1000

Output Format

The compressed String.

Sample Input

```
aaabbccds
```

Sample Output

```
a3b2c2ds
```

Explanation

In the given sample test case 'a' is repeated 3 times consecutively, 'b' is repeated twice, 'c' is repeated twice. But, 'd' and 's' occurred only once that's why we do not write their occurrence.

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

        int length  = s.length();
        int i = 0;
        while(i<length){
            char c = s.charAt(i);
            int count=1;
            if(i == length-1 && c == s.charAt(i-1)){
                break;
            }
            if(i<length-1){
                while(c == s.charAt(i+1)){
                    count++;
                    if(i<length-2){
                        i++;
                    }
                    else{
                        break;
                    }
                }
            }
            System.out.print(c);
            if(count>1){
            System.out.print(count);
            }
            i++;
        }
    }

}
```

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 ans.

Input Format

String

Constraints

Length of String should be between 2 to 1000.

Output Format

String

Sample Input

```
acb
```

Sample Output

```
a2c-1b
```

Explanation

For the sample case, the Ascii code of a=97 and c=99 ,the difference between c and a is 2.Similarly ,the Ascii code of b=98 and c=99 and their difference is -1. So the ans is a2c-1b.

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

    public static void string_diff(String s){
        int length = s.length();

        for(int i=0; i<length-1;i++){
            int diff = (int)s.charAt(i+1) - (int)s.charAt(i);
            System.out.print(s.charAt(i));
            System.out.print(diff);
        }
        System.out.print(s.charAt(length-1));
    }
}
```

A Good String is a string which contains only vowels (a,e,i,o,u) . Given a string S, print a single positive integer N where N is the length of the longest substring of S that is also a Good String.

Note: The time limit for this problem is 1 second, so you need to be clever in how you compute the substrings.

Input Format

A string 'S'

Constraints

Length of string < 10^5

Output Format

A single positive integer N, where N is the length of the longest sub-string of S that is also a Good String.

Sample Input

```
cbaeicde
```

Sample Output

```
3
```

Explanation

Longest good substring is "aei"

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

        int Max = 0;
        for (int i = 0; i < length; i++) {
            char ch = s.charAt(i);
            int count = 0;
            int j_last = 0;
            if (is_vowel(s.charAt(i))) {
                for (int j = i; j < length; j++) {
                    if(is_vowel(s.charAt(j))){
                        count++;
                        j_last = j;
                    }
                    else {
                        break;
                    }
                }
                i=i+(j_last-i);
                Max = Math.max(Max, count);
                }
            }
        System.out.println(Max);
    }

    public static boolean is_vowel(char ch) {
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')
{
            return true;
        }
        return false;
    }
}
```

Take as input N, the size of array. Take N more inputs and store that in an array. Take as input M, the size of second array and take M more inputs and store that in second array. Write a function that returns the sum of two arrays. Print the value returned.

Input Format

Constraints

Length of Array should be between 1 and 1000.

Output Format

Sample Input

```
4
1 0 2 9
5
3 4 5 6 7
```

Sample Output

```
3, 5, 5, 9, 6, END
```

Explanation

Sum of [1, 0, 2, 9] and [3, 4, 5, 6, 7] is [3, 5, 5, 9, 6] and the first digit represents carry over , if any (0 here) .

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

        Scanner sc = new Scanner(System.in);
        int len_a1 = sc.nextInt();
        int [] a1 = new int[len_a1];
        for(int i=0; i<len_a1; i++){
            a1[i] = sc.nextInt();
        }

        int len_a2 = sc.nextInt();
        int [] a2 = new int[len_a2];
        for(int i=0; i<len_a2; i++){
            a2[i] = sc.nextInt();
```

```
        }

        Array_sum(a1,a2);
    }


    public static void Array_sum(int [] arr1, int [] arr2) {
        int length_arr1 = arr1.length-1;
        int length_arr2 = arr2.length-1;

        ArrayList<Integer> list = new ArrayList<>();
        int carry = 0;

        while(length_arr1 >= 0 && length_arr2 >= 0) {
            int sum = 0;
            sum = arr1[length_arr1]+arr2[length_arr2] + carry;
            list.add(sum%10);
            carry = sum/10;
            length_arr1--;
            length_arr2--;
        }

        while(length_arr1 >= 0) {
            int sum = 0;
            sum = arr1[length_arr1] + carry;
            list.add(sum%10);
            carry = sum/10;
            length_arr1--;
        }

        while(length_arr2 >= 0) {
            int sum = 0;
            sum = arr2[length_arr2] + carry;
            list.add(sum%10);
            carry = sum/10;
            length_arr2--;
        }

        if(carry != 0 ) {
            list.add(carry);
        }

        for(int i=list.size()-1; i >= 0; i--) {
            System.out.print(list.get(i) + ", ");
        }
        System.out.print("END");
    }}
```

You are provided an array of numbers. You need to arrange them in a way that yields the largest value.

Input Format

First line contains integer t which is number of test case.
For each test case, you are given a single integer n in the first line which is the size of array A[] and next line contains n space separated integers denoting the elements of the array A .

Constraints

1<=t<=100

1<=m<=100

1<=A[i]<=10^5

Output Format

Print the largest value.

Sample Input

```
1
4
54 546 548 60
```

Sample Output

```
6054854654
```

Explanation

Upon rearranging the elements of the array , 6054854654 is the largest possible number that can be generated.

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

```
        while (t > 0) {

            int n = sc.nextInt();
            int[] arr = new int[n];
            for (int i = 0; i < arr.length; i++) {
                arr[i] = sc.nextInt();
            }
            Sort(arr);
            for (int i = 0; i < arr.length; i++) {
                System.out.print(arr[i]);

            }
            System.out.println();
            t--;
        }

    }

    public static void Sort(int[] arr) {

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

            for (int j = 0; j < arr.length -i; j++) {
                if (compare(arr[j], arr[j + 1])) {
                    int t = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = t;
                }

            }

        }

    }

    public static boolean compare(int n1, int n2) {
        String s1 = "" + n1 + n2;
        String s2 = "" + n2 + n1;
        if (Long.parseLong(s2) > Long.parseLong(s1)) {
            return true;
        }
        return false;

    }

}
```

Take as input S, a string. Write a function that removes all consecutive duplicates. Print the value returned.

Input Format

String

Constraints

A string of length between 1 to 1000

Output Format

String

Sample Input

```
aabccba
```

Sample Output

```
abcba
```

Explanation

For the given example, "aabccba", Consecutive Occurrence of a is 2, b is 1, and c is 2.

After removing all of the consecutive occurences, the Final ans will be : - "abcba".

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

    public static void remove_duplicates(String s){
        int length = s.length();

        for(int i = 0; i<length-1; i++){
            if(s.charAt(i) != s.charAt(i+1)){
                System.out.print(s.charAt(i));
            }
        }
            System.out.print(s.charAt(length-1));
    }
}
```

Take as input a 2-d array. Print the 2-D array in spiral form anti-clockwise.

Input Format

Two integers M(row) and N(colomn) and further M * N integers(2-d array numbers).

Constraints

Both M and N are between 1 to 10.

Output Format

All M * N integers separated by commas with 'END' written in the end(as shown in example).

Sample Input

```
4 4
11 12 13 14
21 22 23 24
31 32 33 34
41 42 43 44
```

Sample Output

```
11, 21, 31, 41, 42, 43, 44, 34, 24, 14, 13, 12, 22,
```

Explanation

For spiral level anti-clockwise traversal, Go for first column-> last row ->last column-> first row and then do the same traversal for the remaining matrix .

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

        int row_length = sc.nextInt();
        int col_length = sc.nextInt();

        int [] [] matrix = new int [row_length] [col_length];

        for(int i=0; i<row_length; i++){
            for (int j=0; j<col_length; j++){
                matrix[i][j] = sc.nextInt();
            }
        }

        anti_spiral(matrix);
        System.out.print("END");
    }

    public static void anti_spiral(int [] [] matrix){
        int min_row = 0, min_col = 0;
        int max_row = matrix.length-1, max_col = matrix[0].length-1;
        int total_elements = matrix.length * matrix[0].length;
        int count = 0;

        while(count < total_elements){

            for(int i = min_row; i <= max_row; i++){
                if (count < total_elements){
                    System.out.print(matrix[i][min_col]+", ");
                    count++;
                }
            }
            min_col++;

            for(int j = min_col; j <= max_col; j++){
                if (count < total_elements){
                    System.out.print(matrix[max_row][j]+", ");
                    count++;
                }
            }
            max_row--;
```

```
            for(int k = max_row; k >= min_row; k--){
                if (count < total_elements){
                    System.out.print(matrix[k][max_col]+", ");
                    count++;
                }
            }
            max_col--;

            for(int l = max_col; l >= min_col; l--){
                if (count < total_elements){
                    System.out.print(matrix[min_row][l]+", ");
                    count++;
                }
            }
            min_row++;
        }
    }
}
```

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

Constraints

Length of string should be between 1 to 1000.

Output Format

String

Sample Input

```
abcg
```

Sample Output

```
badf
```

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

    public static void replace_char(String s){
        int length = s.length();

        for(int i=0; i<length; i++){
            int i_value = (int)s.charAt(i);
            if(i%2==0){
                i_value+=1;
            }
            else{
                i_value-=1;
            }
            System.out.print((char)i_value);
        }
    }
}
```

Given an n x m matrix, where every row and column is sorted in increasing order, and a number x . Find if element x is present in the matrix or not.

Input Format

First line consists of two space separated integers N and M, denoting the number of element in a row and column respectively. Second line of each test case consists of N*M space separated integers denoting the elements in the matrix in row major order. Third line of each test case contains a single integer x, the element to be searched.

Constraints

```
1 <= N,M <= 30 0 <= A[i] <= 100
```

Output Format

Print 1 if the element is present in the matrix, else 0.

Sample Input

```
3 3
3 30 38
44 52 54
57 60 69
62
```

Sample Output

```
0
```

Explanation

Search the element in the sorted matrix. If the element is present print 1 otherwise print 0. In the sample input,in first case 62 is not present in the matrix so 0 is printed. Similarly, for second case 55 is present in the matrix so 1 is printed

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

        int row_length = sc.nextInt();
        int col_length = sc.nextInt();

        int [] [] matrix = new int [row_length] [col_length];

        for(int i=0; i<row_length; i++){
            for (int j=0; j<col_length; j++){
                matrix[i][j] = sc.nextInt();
            }
        }

        int target = sc.nextInt();

        System.out.print(searchMatrix(matrix,target));
    }


    public static int searchMatrix(int[][] matrix, int target) {
        int m = matrix.length;
        int n = matrix[0].length;

        int row = 0;
        int col = n - 1;

        while (row < m && col > -1) {
            if (matrix[row][col] == target) {
                return 1;
            } else if (matrix[row][col] < target) {
                row++;
            } else {
                col--;
            }
        }
        return 0;
    }

}
```

Take as input S, a string. Write a function that returns true if the string is a palindrome and false otherwise. Print the value returned.

Input Format

String

Constraints

String length between 1 to 1000 characters

Output Format

Boolean

Sample Input

```
abba
```

Sample Output

```
true
```

Explanation

A string is said to be palindrome if reverse of the string is same as string. For example, "abba" is palindrome as it's reverse is "abba", but "abbc" is not palindrome as it's reverse is "cbba".

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();
        System.out.print(pallindrome(s));
    }

    public static boolean pallindrome(String s){
        int i=0;
        int j = s.length()-1;
        while(i<j){
            if(s.charAt(i) != s.charAt(j)){
                return false;
            }
            i++;
            j--;
        }
        return true;
    }
}
```

Take as input S, a string. Write a function that returns the character with maximum frequency. Print the value returned.

Input Format

String

Constraints

A string of length between 1 to 1000.

Output Format

Character

Sample Input

```
aaabacb
```

Sample Output

```
a
```

Explanation

For the given input string, a appear 4 times. Hence, it is the most frequent character.

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

        System.out.print(s.charAt(max_frequency(s)));
    }

    public static int max_frequency(String s){
        int length = s.length();
        int max=0;
        int i = 0;
        int count = 1;
        int max_index = 0;
        while(i < length-1){
            if(s.charAt(i) != s.charAt(i+1)){
                if(max < count){
                    max = count;
                    max_index = i;
                }
                count=0;
            }
            if(i==length-2){
                if(s.charAt(i) == s.charAt(i+1)){
                    count++;
                }
                if(max < count){
                    max_index = i;
                }
            }
            count++;
            i++;
        }
        return max_index;
    }
}
```

Take as input a 2-d array.Print the 2-D array in sprial form clockwise.

Input Format

Two integers M(row) and N(colomn) and further M * N integers(2-d array numbers).

Constraints

Both M and N are between 1 to 10.

Output Format

All M * N integers separated by commas with 'END' written in the end(as shown in example).

Sample Input

```
4 4
11 12 13 14
21 22 23 24
31 32 33 34
41 42 43 44
```

Sample Output

```
11, 12, 13, 14, 24, 34, 44, 43, 42, 41, 31, 21, 22,
```

Explanation

For spiral level clockwise traversal, Go for first row-> last column ->last row -> first column and then do the same traversal for the remaining matrix .

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

        int row_length = sc.nextInt();
        int col_length = sc.nextInt();

        int [] [] matrix = new int [row_length] [col_length];

        for(int i=0; i<row_length; i++){
            for (int j=0; j<col_length; j++){
                matrix[i][j] = sc.nextInt();
            }
        }

        spiral(matrix);
        System.out.print("END");

    }

    public static void spiral(int[][]arr) {
        int min_row=0;
        int min_col=0;
        int max_row=arr.length-1;
        int max_col=arr[0].length-1;
        int total_element=arr.length*arr[0].length;
        int count=0;

        while(count<total_element) {
            for(int i=min_col; i<=max_col; i++) {
                if(count<total_element) {
                    System.out.print(arr[min_row][i]+", ");
                    count++;
                }
            }

            min_row++;
            for(int j=min_row; j<=max_row; j++) {
```

```
            if(count<total_element) {
                System.out.print(arr[j][max_col]+", ");
                count++;
            }
        }

        max_col--;
        for(int r = max_col; r>=min_col; r--) {
            if(count<total_element) {
                System.out.print(arr[max_row][r]+", ");
                count++;
            }
        }

        max_row--;
        for(int k=max_row; k>=min_row; k--) {
            if(count<total_element) {
                System.out.print(arr[k][min_col]+", ");
                count++;
            }
        }
        min_col++;
        }
    }

}
```

Given a 2D array of size N x N. Rotate the array 90 degrees anti-clockwise.

Input Format

First line contains a single integer N. Next N lines contain N space separated integers.

Constraints

N < 1000

Output Format

Print N lines with N space separated integers of the rotated array.

Sample Input

```
4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

Sample Output

```
4 8 12 16
3 7 11 15
2 6 10 14
1 5 9 13
```

Explanation

Rotate the array 90 degrees anticlockwise.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
 Scanner sc = new Scanner(System.in);
        int m = sc.nextInt();
        int [] [] a = new int[m][m];
        for(int i=0; i<m; i++){
            for(int j=0; j<m; j++){
                a[i][j] = sc.nextInt();
            }
        }
    Transpose(a);

    for(int i=m-1; i>-1; i--){
        for(int j=0; j<m; j++){
            System.out.print(a[i][j]+" ");
        }
        System.out.println();
    }
    }

    public static void Transpose(int [] [] a){
        int row_length = a.length;
        int col_length = a[0].length;
        for(int i=0; i<row_length; i++){
            for(int j=i+1; j<col_length; j++){
                int temp = a[i][j];
                a[i][j] = a[j][i];
                a[j][i] = temp;
            }
        }
    }

}
```

Take as input S, a string. Write a program that gives the count of substrings of this string which are palindromes and Print the ans.

Input Format

Single line input containing a string

Constraints

Length of string is between 1 to 1000.

Output Format

Integer output showing the number of palindromic substrings.

Sample Input

```
abc
```

Sample Output

```
3
```

Explanation

For the given sample case , the palindromic substrings of the string abc are "a","b" and "c".So, the ans is 3.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();
        int count=0;
        int length = s.length();

        int check_length = 1;
        while(check_length <= length){
            for(int i=0; i+check_length<=length; i++){
                if(pallindrome(s.substring(i,i+check_length))){
                    count++;
                }
            }
            check_length++;
        }
        System.out.print(count);
    }

      public static boolean pallindrome(String s){
        int i=0;
        int j = s.length()-1;
        while(i<j){
            if(s.charAt(i) != s.charAt(j)){
                return false;
            }
            i++;
            j--;
        }
        return true;
    }
}
```

Given a square matrix, print its transpose.

Input Format

First line contains the N ,size of the square matrix. Next N lines contains N integers each denoting the elements of the matrix

Constraints

```
1 <= N <= 10^3
```

Output Format

Print N lines each containing N elements. These are the elements of the new matrix.

Sample Input

```
5
1 46 4 60 100
28 52 97 80 59
6 33 62 42 12
57 31 56 89 47
1 50 73 53 99
```

Sample Output

```
1 28 6 57 1
46 52 33 31 50
4 97 62 56 73
60 80 42 89 53
100 59 12 47 99
```

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

Scanner sc = new Scanner(System.in);
        int m = sc.nextInt();
        int [] [] a = new int[m][m];
        for(int i=0; i<m; i++){
            for(int j=0; j<m; j++){
                a[i][j] = sc.nextInt();
            }
        }
    Transpose(a);

    for(int i=0; i<m; i++){
        for(int j=0; j<m; j++){
            System.out.print(a[i][j]+" ");
        }
        System.out.println();
    }
    }

    public static void Transpose(int [] [] a){
        int row_length = a.length;
        int col_length = a[0].length;
        for(int i=0; i<row_length; i++){
            for(int j=i+1; j<col_length; j++){
                int temp = a[i][j];
                a[i][j] = a[j][i];
                a[j][i] = temp;
            }
        }
    }
}
```

Given a M x N matrix consisting of only 0 or 1, change all elements of row i and column j to 0 if cell (i, j) has value 0. Do this without using any extra space for every (i, j) having value 0.

Input Format

Two integers M and N denoting number of rows and columns of the matrix
M * N elements of the matrix.

Constraints

```
0<=N<=50
0<=mat[i][j]<=1
```

Output Format

Formatted matrix.

Sample Input

```
5 5
1 1 0 1 1
1 1 1 1 1
1 1 1 0 1
1 1 1 1 1
0 1 1 1 1
```

Sample Output

```
0 0 0 0 0
0 1 0 0 1
0 0 0 0 0
0 1 0 0 1
0 0 0 0 0
```

Explanation

0's are present at (0, 2), (4, 0), and (2, 3) in the input matrix.
So, we change all elements of the following cells to 0.

row 0 and column 2
row 4 and column 0
row 2 and column 3

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

        int row_length = sc.nextInt();
        int col_length = sc.nextInt();

        int [] [] matrix = new int [row_length] [col_length];
        int count=0;
        for(int i=0; i<row_length; i++){
            for (int j=0; j<col_length; j++){
                matrix[i][j] = sc.nextInt();
                if(matrix[i][j] == 0){
                    count++;
                }
            }
        }

        change_elements(matrix,zero_index(matrix,count));
       // display(zero_index(matrix,count));
    }

    public static  int [] [] zero_index(int [] [] matrix, int zeros){
        int row_length = matrix.length;
        int col_length = matrix[0].length;


        int [][] row_index = new int [zeros][2];

        int col_i =0;
        for(int i=0; i<row_length; i++){
            for(int j=0; j<col_length; j++){
                if(matrix[i][j] == 0){
                    int [] col_index = new int [2];
                    col_index[1] = j;
                    col_index[0] = i;
                    row_index[col_i] = col_index;
                    col_i++;
                }
            }
        }
        return row_index;
    }
```

```
    public static void change_elements(int [] [] matrix, int [] []
zero_index){
        int row_length = matrix.length;
        int col_length = matrix[0].length;

        for(int i=0; i<row_length; i++){
            for(int j=0; j<col_length; j++){
                for(int k=0; k<zero_index.length; k++){
                    if(i == zero_index[k][0] || j == zero_index[k][1]){
                        matrix[i][j] = 0;
                    }
                }
            }
        }

        display(matrix);

    }

    public static void display(int [] [] matrix){
        int row_length = matrix.length;
        int col_length = matrix[0].length;

        for(int i=0; i<row_length; i++){
            for(int j=0; j<col_length; j++){
                System.out.print(matrix[i][j]+" ");
            }
            System.out.println();
        }
    }
}
```

Take as input a two-d array. Wave print it row-wise.

Input Format

Two integers M(row) and N(column) and further M * N integers(2-d array numbers).

Constraints

Both M and N are between 1 to 10.

Output Format

All M * N integers are seperated by commas with 'END' written in the end(as shown in example).

Sample Input

```
4 4
11 12 13 14
21 22 23 24
31 32 33 34
41 42 43 44
```

Sample Output

```
11, 12, 13, 14, 24, 23, 22, 21, 31, 32, 33, 34, 44,
```

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

        int row_length = sc.nextInt();
        int col_length = sc.nextInt();

        int [] [] matrix = new int [row_length] [col_length];

        for(int i=0; i<row_length; i++){
            for (int j=0; j<col_length; j++){
```

```
                matrix[i][j] = sc.nextInt();
            }
        }

        wave_print_row(matrix);
        System.out.print("END");
    }

    public static void wave_print_row(int [] [] matrix){
        int min_row = 0;
        int max_col = matrix[0].length-1;

        int count = 0;
        int total_elements = matrix.length * matrix[0].length;

        while(count < total_elements){

            for(int i = 0; i <= max_col; i++){
                if(count < total_elements){
                    System.out.print(matrix[min_row][i]+", ");
                    count++;
                }
            }
            min_row++;

            for(int j = max_col; j >= 0; j--){
                if(count <  total_elements){
                    System.out.print(matrix[min_row][j]+", ");
                    count++;
                }
            }
            min_row++;
        }
    }
}
```

Take as input a two-d array. Wave print it column-wise.

Input Format

Two integers M(row) and N(colomn) and further M * N integers(2-d array numbers).

Constraints

Both M and N are between 1 to 10.

Output Format

All M * N integers seperated by commas with 'END' wriiten in the end(as shown in example).

Sample Input

```
4 4
11 12 13 14
21 22 23 24
31 32 33 34
41 42 43 44
```

Sample Output

```
11, 21, 31, 41, 42, 32, 22, 12, 13, 23, 33, 43, 44,
```

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

        int row_length = sc.nextInt();
        int col_length = sc.nextInt();

        int [] [] matrix = new int [row_length] [col_length];
```

```
        for(int i=0; i<row_length; i++){
            for (int j=0; j<col_length; j++){
                matrix[i][j] = sc.nextInt();
            }
        }

        wave_print_column(matrix);
        System.out.print("END");
    }

    public static void wave_print_column(int [] [] matrix){
        int min_col = 0;
        int max_row = matrix.length-1;

        int count = 0;
        int total_elements = matrix.length * matrix[0].length;

        while(count < total_elements){

            for(int i = 0; i <= max_row; i++){
                if(count < total_elements){
                    System.out.print(matrix[i][min_col]+", ");
                    count++;
                }
            }
            min_col++;

            for(int j = max_row; j >= 0; j--){
                if(count <  total_elements){
                    System.out.print(matrix[j][min_col]+", ");
                    count++;
                }
            }
            min_col++;
        }
    }
}
```