# **Search a number**

Given an n x m matrix, where every row and column is sorted in increasing order, and a number x . The task is to find whether element x is present in the matrix or not.

Input:

The first line of input contains a single integer T denoting the number of test cases. Then T test cases follow. Each test case consists of three lines.

First line of each test case consist 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.

Output:

Corresponding to each test case, print in a new line, 1 if the element x is present in the matrix, otherwise simply print 0.

Constraints:

1<=T<=200

1<=N,M<=30

Sample Input

2

3 3

3 30 38 44 52 54 57 60 69

62

1 6

18 21 27 38 55 67

55

Sample Output:

0

1

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <assert.h>

#include <limits.h>

#include <stdbool.h>

int main() {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

int a[100],i,j,k,n,lock,e;

scanf("%d",&n);

while(n--){

lock=0;

scanf("%d %d",&i,&j);

i=i\*j;

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

scanf("%d",&a[k]);

}

scanf("%d",&e);

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

if(e==a[k])lock=1;

}

if(lock==1)printf("1\n");

else printf("0\n");

}

return 0;

}

# **Magic Square**

A magic square is a simple mathematical game developed during the 1500. Square is divided into equal number of rows and columns. Start filling each square with the number from 1 to num ( where num = No of Rows X No of Columns ). You can only use a number once. Fill each square so that the sum of each row is the same as the sum of each column. In the example shown here, the sum of each row is 15, and the sum of each column is also 15.

Enter a 3X3 matrix check whether it is magic square or not.Print "magic square" if it is magic square else print "not magic sqaure".

**Sample Input**

4 9 2 3 5 7 8 1 6

**Sample Output**

magic square

**Sample Input**

4 5 7 8 2 1 0 9 3

**Sample Output**

not magic square

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <assert.h>

#include <limits.h>

#include <stdbool.h>

int main() {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

int i,j,a[1000][1000],s1=0,s2=0,f=0;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

scanf("%d",&a[i][j]);

}

}

for(i=0;i<3;i++)

{

s1=0;

for(j=0;j<3;j++)

{

s1=s1+a[i][j];

}}

for(i=0;i<3;i++)

{

s2=0;

for(j=0;j<3;j++)

s2=s2+a[j][i];

if(s1!=s2)

{

f=1;break;

}

}

if(f==1)

printf("not magic square");

else

printf("magic square");

return 0;

}

# **Matrix in Spiral Form**

Given a 2D matrix of size MXN, print it in spiral form. See the following examples.

Sample Input:

4 4

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

Sample Output:

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

Sample Input:

3 6

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

Sample Output:

1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11

#include <stdio.h>

void spiralPrint(int m, int n, int a[1000][1000])

{

int i, k = 0, l = 0;

while (k < m && l < n)

{

for (i = l; i < n; ++i)

{

printf("%d ", a[k][i]);

}

k++;

for (i = k; i < m; ++i)

{

printf("%d ", a[i][n-1]);

}

n--;

if ( k < m)

{

for (i = n-1; i >= l; --i)

{

printf("%d ", a[m-1][i]);

}

m--;

}

if (l < n)

{

for (i = m-1; i >= k; --i)

{

printf("%d ", a[i][l]);

}

l++;

}

}

}

int main()

{

int n,m,i,j;

int a[1000][1000];

scanf("%d %d",&n,&m);

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

for(j=0;j<m;j++) scanf("%d",&a[i][j]);

}

spiralPrint(n,m, a);

return 0;

}

# **Addition of submatrix**

Given a matrix of size n x m. You are given a series of submatrix inside the matrix. Find the sum of all elements inside that submatrix. Submatrix position are given in terms of its: top\_left\_cell and bottom\_right\_cell.

Input:

The first line of input contains an integer T denoting the number of test cases.

The first line of each test case is n and m,n is the number of rows and m is the number of columns.

The second line of each test case contains C[n][m].

The third line contains four value of x1, y1, x2, y2.

x1, y1 is the top left cell and x2, y2 is the bottom right cell.

Output:

Print the sum of all elements inside that submatrix.

Constraints:

1 ≤ T ≤ 15

1 ≤ n,m ≤ 10

1 ≤ C[n][m] ≤ 1000

Example:

Input:

2

5 6

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

3 4 4 5

3 3

9 8 7 4 2 1 6 5 3

1 2 3 3

Output:

78

26

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <assert.h>

#include <limits.h>

#include <stdbool.h>

int main() {

int i,j,a[1000][1000],x1,x2,y1,y2,sum,n,r,c;

scanf("%d",&n);

while(n--){

sum=0;

scanf("%d %d",&r,&c);

for(i=0;i<r;i++){

for(j=0;j<c;j++)scanf("%d",&a[i][j]);

}

scanf("%d %d %d %d",&x1,&y1,&x2,&y2);

for(i=x1-1;i<x2;i++){

for(j=y1-1;j<y2;j++)sum=sum+a[i][j];

}

printf("%d\n",sum);

}

return 0;

}

# **Sum of elements in a matrix**

Find sum of all elements in a matrix except the elements in row and/or column of given cell?

Given a 2D matrix and a set of cell indexes e.g., an array of (i, j) where i indicates row and j column. For every given cell index (i, j), find sums of all matrix elements except the elements present in i’th row and/or j’th column.

Example:

mat[][] = { {1, 1, 2}

{3, 4, 6}

{5, 3, 2} }

Array of Cell Indexes: {(0, 0), (1, 1), (0, 1)}

Output: 15, 10, 16

Sample Input

3 3

1 1 2

3 4 6

5 3 2

0 0

Sample Output

26

Explanation: first line of input indicates N rows,M columns;Second line indicates NXM matrix and last line indicates row and column to exclude from sum.

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <assert.h>

#include <limits.h>

#include <stdbool.h>

int main() {

int i,j,sum=0,r,c,a[100][100],x,y;

scanf("%d %d",&r,&c);

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

for(i=0;i<r;i++){

for(j=0;j<c;j++)scanf("%d",&a[i][j]);

}

scanf("%d %d",&x,&y);

for(i=0;i<r;i++){

for(j=0;j<c;j++){

if(i!=x||j!=y)sum=sum+a[i][j];

}

}

printf("%d",sum);

return 0;

}**Sort a matrix**

You have been given a NXN matrix,Display the matrix by sorting in ascending order.

Sample Input

3 3

1 3 4

2 5 8

6 7 9

Sample Output

1 2 3

4 5 6

7 8 9

**YOUR ANSWER**

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <assert.h>

#include <limits.h>

#include <stdbool.h>

int main() {

int i,j,a[100],r,c,lock,l,in,small;

scanf("%d %d",&r,&c);

for(i=0;i<r\*c;i++){

scanf("%d",&a[i]);

}

for(i=0;i<r\*c;i++){

small=a[i];lock=0;

for(j=i;j<r\*c;j++)

{

if(small>a[j]){

lock=1;

l=j;

small=a[j];

}

}

if(lock==1){

int x=a[i];

a[i]=a[l];

a[l]=x;

}

}

in=0;

for(i=0;i<r;i++){

for(j=0;j<c;j++){

printf("%d ",a[in]);in++;

}printf("\n");

}

return 0;

}

# **Kth element in Matrix**

Given an n x n matrix, where every row and column is sorted in non-decreasing order.

Find the kth smallest element in the given 2D array.Your task is to complete the

function kth Smallest which takes two arguments the first is argument is the order of

the matrix (n) and the function returns the kth smallest element in the matrix.

For example, consider the following 2D array.

10, 20, 30, 40

15, 25, 35, 45

24, 29, 37, 48

32, 33, 39, 50

The 3rd smallest element is 20 and 7th smallest element is 30

Input:

The first line of input is an integer T denoting the no of test cases .

Then T lines follow . The first line of each test case contains an integer

N denoting the size of the matrix then in the next line are N\*N space separated values of the matrix .

The third line contains an integer k.

Output:

For each test case print in a new line the kth smallest element of the matrix .

Constraints:

1<=T<=100

1<= N<=20

1<= M[][]<=100

Example

Input

1

4

16 28 60 64 22 41 63 91 27 50 87 93 36 78 87 94

3

Output

27

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <assert.h>

#include <limits.h>

#include <stdbool.h>

int main() {

int n,i,j,a[1000],small,lock,l,x,y,t;

scanf("%d",&t);

while(t--){

scanf("%d",&n);

for(i=0;i<n\*n;i++)scanf("%d",&a[i]);

scanf("%d",&y);

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

lock=0;small=a[i];

for(j=i;j<n\*n;j++){

if(small>a[j]){

lock=1;

l=j;

small=a[j];

}

}

if(lock==1){

x=a[i];

a[i]=a[l];

a[l]=x;

}

}

printf("%d\n",a[y-1]);}

return 0;

}

# **Return length of String**

U have been given the task to return the count of Alphabets in a string.

A string can be entered in lowercase ,uppercase or mixed case

Sample Input

Hello Everbody class123

Sample Output

18

int b=0,i;

for(i=0;i<strlen(a);i++){

if(((int)a[i]<48||(int)a[i]>57)&&a[i]!=' ')b++;

}

return b;

}

# 

# **Copying String**

U have been given the task to copy the string from source to destination.if the source character is lowercase copy it in uppercase into destination,and if it is uppercase copy it in lowercase

Sample Input

Programming Can Be Fun

Sample output

pROGRAMMING cAN bE fUN

int i,x;

for(i=0;a[i]!='\0';i++){

x=(int)a[i];

if(x>=65&&x<=90) b[i]=(char)x+32;

else if(x>=97&&x<=122) b[i]=(char)x-32;

else b[i]=a[i];

}

}

int main()

{

char a[100],b[100];

scanf("%[^\n]",a);

myCopy(a,b);

printf("%s\n",b);

}

# **Comparing Strings**

Problem statement

strcmp function in C compares two strings. It return 0 if both string are equal.It returns a positive value, if string A is lexicographically smaller than string B, else returns a negative number. Write your own implementation of strcmp function in C.

Input

You need to write the body of a function with signature: myStrcpy(char \*str1,char \*str2); str1 and str2 will be a pointer to the two character array that will store the strings.

Output

Your function must return 0 if the strings are equal. Else your function must return the difference between the ASCII value of characters ( str1[i] - str2[i], for some i )at index at which the strings differ.

**YOUR ANSWER**

#include<stdio.h>

#define MAX 100

int myStrcmp(char\* str1,char \*str2)

{

int r;

r=strcmp(str1,str2);

return r;

}

int main()

{

int t,n;

char str1[MAX],str2[MAX];

scanf("%d",&t);

while(t--)

{

scanf(" %s",str1);

scanf(" %s",str2);

printf("%d\n",myStrcmp(str1,str2));

}

return 0;

}

# **Anagram**

Given two words, verify whether they are anagram or not. Two strings are anagrams if they are rearrangements of the same set of letters. for example , the words 'creative'and 'reactive' have same set of letters, so they are anagrams.

**Input**

First line of input will contain a number T = number of test cases. Next T lines will contain two words each separated by a space.

**Output**

For each pair of words, print "YES", if they are anagrams, else print "NO"

# include <stdio.h>

# define len 256

int areAnagram(char str1[], char str2[])

{

int l1=strlen(str1),l2=strlen(str2),i,l=0,j,lock;

if(l1==l2){

for(i=0;i<l1;i++){

lock=0;

for(j=0;j<l2;j++){

if(str1[i]==str2[j])lock=1;

}

if(lock==0)l=l-1000;

else l=l+1;

}

if(l>0){

for(i=0;i<l2;i++){

lock=0;

for(j=0;j<l1;j++){

if(str2[i]==str1[j])lock=1;

}

if(lock==0)l=l-1000;

else l=l+1;

}

}

if(l>0) return 1;

else return 0;

}else return 0;

}

int main()

{

int t;

char str1[len];

char str2[len];

scanf("%d",&t);

while(t--)

{

scanf(" %s",str1);

scanf(" %s",str2);

if ( areAnagram(str1, str2) )

printf("anagram\n");

else

printf("not anagram\n");

}

return 0;

}

# **Cocatenation of Strings**

**Problem statement**

strcat function in C concatenates two strings. Write your own implementation of this function.

**Input**

2

Hello

world

Programming can be fun

If put in right efforts

**Output**

Hello world

Programming can be fun If put in right efforts

Explanation

2 is number of test cases

We need to concatenate second string to first string but we should have space between first string and second string during concatenation

# include <stdio.h>

# define len 256

void myStrcat(char dst[], char src[])

{

char str1[len];

strcat(dst ," ");

strcat(dst,src);

}

int main()

{

int t;

char str1[len];

char str2[len];

scanf("%d",&t);

while(t--)

{

scanf(" %[^\n]",str1);

scanf(" %[^\n]",str2);

myStrcat(str1,str2);

printf("%s\n",str1);

}

return 0;

}

# **Maximum Occuring Character**

iven a string, find the maximum occurring character in the string. If more than one character occurs maximum number of time

then print the lexicographically smaller character.

Input:

The first line of input contains an integer T denoting the number of test cases.

Each test case consist of a string in 'lowercase' only in a separate line.

Output:

Print the lexicographically smaller character which occurred the maximum time.

Constraints:

1 ≤ T ≤ 30

1 ≤ |s| ≤ 100

Example:

**Sample Input:**

2

testsample

geeksforgeeks

**Sample Output:**

e

e

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <assert.h>

#include <limits.h>

#include <stdbool.h>

int main() {

int t,i,j,c,large,idd,id;

scanf("%d",&t);

while(t--){

char a[1000];

scanf("%s",&a);

large=0;

for(i=0;i<strlen(a);i++){

char x=a[i];c=0;

for(j=0;j<strlen(a);j++){

if(x==a[j]){

c++;id=j;

}

}

if(large<c){large=c;idd=i;}

}

printf("%c\n",a[idd]);

}

return 0;

}

# **Simple Encryption**

**Problem statement**

Caeser cipher is an age old encryption technique that was used by Roman emperor Julius Caeser to write letters to his queen Cleopatra. Given a string encrypted using Caeser Cipher, find the plain text.

**Input**

You will be given a string comprising of only upper English letters. In caeser cipher each letter of plain text is substituted by another letter that is some key units ahead of it towards right in a circular traversal. For example, if key is 3, A becomes D, B becomes E and Z becomes C. First line of input will contain a number T = no. of test cases. Each test case will contain two lines. First line will contain the key. Next line will an encrypted string comprising of only upper case letters.

Output

For each string, print the original plain text on a single line.

**Sample Input**

3

3

DEF

2

JGNNQ

1

NZDPEFTDIPPM

**Sample Output**

ABC

HELLO

MYCODESCHOOL

#include<stdio.h>

void myEncryption(char s[],int n)

{

int i,d;

for(i=0;i<strlen(s);i++){

int c=(int)s[i];

if(c-n<65){d=65+c-n;d=90+d;}

else {d=c-n;}

s[i]=(char)d;

}}

#include<string.h>

int main()

{

int t,n;

char str1[1000];

scanf("%d",&t);

while(t--)

{

scanf("%d",&n);

scanf(" %s",str1);

myEncryption(str1,n);

printf("%s\n",str1);

}

return 0;

}

# **Verify Palindrome**

Problem statement

Verify whether a given string is palindrome or not

Input

First line of test case will contain an integer T = no. of test cases. Following T lines will contain one word each.

Output

For each test case, print "YES", if the given word is a palindrome. Else, print "NO"

Sample Input

4

madam

malayalam

blob

eve

Sample Output

YES

YES

NO

YES

#include<stdio.h>

#define len 100

int isPalindrome(char str[])

{

char a[100];

int i,j=strlen(str)-1,lock=0;

for(i=0;i<strlen(str);i++){if(str[i]==str[j])lock=lock+1;

else lock=lock-100;

j--;

}

if(lock>0)return 1;

else return 0;

}

int main()

{

int t,n;

char str1[len];

scanf("%d",&t);

while(t--)

{

scanf(" %s",str1);

if(isPalindrome(str1))

printf("YES\n",str1);

else

printf("NO\n",str1);

}

return 0;

}

# **Check if string is sub-sequence**

Problem statement

Given two strings , find whether one string is sub-sequence of another string. sub-sequence of a string is a string that can be obtained by deleting zero or more characters from the original string.

Input

First line of input will contain an integer T = number of test cases. Next T lines will each contain two alphanumeric strings separated by space.

Output

For each test case, print "YES" if one of the strings is sub-sequence of another string. Else, print "NO". String A is sub-sequence of string B if A can be obtained by deleting zero or more characters from B.

Sample Input

5

bangalore blr

Hello hel

hey harvey

Tsunami sun

building gun

Sample Output

YES

NO

YES

YES

NO

#include<stdio.h>

#define LEN 100

int isSubSequence(char str1[], char str2[])

{

int i,j,l=0,lock,t=0;

char a;

if(strlen(str2)<strlen(str1)){for(i=0;i<strlen(str2);i++){

a=str2[i];lock=0;

for(j=t;j<strlen(str1);j++){

if(a==str1[j]){lock=1;t=j;break;}

}

if(lock==1)l=l+1;

else l=l-1000;}}

else{

for(i=0;i<strlen(str1);i++){

a=str1[i];lock=0;

for(j=t;j<strlen(str2);j++){

if(a==str2[j]){lock=1;t=j;break;}

}

if(lock==1)l=l+1;

else l=l-1000;

}}

if(l>0)return 1;

else return 0;}

#include<string.h>

int main()

{

char str1[LEN];

char str2[LEN];

int t,m,n;

scanf("%d",&t);

while(t--)

{

scanf(" %s",str1);

scanf("%s",str2);

if(isSubSequence(str1,str2))

printf("YES\n");

else

printf("NO\n");

}

return 0;

}

# **Removing Duplicates From Sorted String**

Problem statement

Given a sorted string comprising of only upper case English letters, remove all the duplicate letters from the string. For example, AAAABBCCDD is a sorted string in which alphabets are in increasing order from A to Z and will become ABCD after duplicates are removed.

Input

You need to write the body of a function with signature: void RemoveDuplicates(char \*C); C will be a pointer to the character array that will store the string.

Output

Your method should have no return, but it must remove duplicate characters from any input string. You must not print anything in the function. You only need to modify the given string.

#include<stdio.h>

#include<string.h>

#define MAX 100

void removeDuplicates(char str[])

{

int lock=0,i,j;

char a;

for(i=0;i<strlen(str);){

a=str[i];

for(j=i;str[j]==a;j++)i++;

printf("%c",a);

}

printf("\n");

}

int main()

{

int t;

char str1[MAX];

scanf("%d",&t);

while(t--)

{

scanf(" %s",str1);

removeDuplicates(str1);

}

return 0;

}

# **Reverse order of words in a sentence**

Problem statement

Given a sentence as string, reverse the order of words in the sentence. For example 'all is well' should become, 'well is all'

Input

First line will contain a number T = number of test cases. Following lines will each contain a sentence not larger than 100 characters.

Output

For each sentence, print the sentence with order of words reversed.

Sample Input

3

I am Albert

Are you going?

all is well in hyderabad

Sample Output

Albert am I

going? you Are

hyderabad in well is all

#include<stdio.h>

#include<string.h>

#define MAXLEN 100

void reverseWords(char buffer[], int strLength)

{

int i, j,l,c=0,d;

char b[100];

for(l=0;buffer[l]!='\0';l++);

d=l-1;

for(i=l-1;i>=0;i--){

if(buffer[i]==' '){

for(j=i+1;j<=d;j++,c++){

b[c]=buffer[j];

}

b[c]=buffer[i];

c++;

d=i-1;

}

}

for(i=0;i<=d;i++,c++){

b[c]=buffer[i];}

b[c]='\0';

for(i=0;b[i]!='\0';i++){

buffer[i]=b[i];

} buffer[i]='\0';

}

int main()

{

char string[MAXLEN];

int t;

scanf("%d",&t);

while(t--)

{

scanf(" %[^\n]",string);

reverseWords(string, strlen(string));

puts(string);

}

}