Program to print all distinct elements of given input arrays.

Also

print the total of the distinct elements.

Input:

Arr1 = {1,2,3,4,5}

Arr 2 = {2,6,8,10}

In C Language

```
#include<stdio.h>

int Not_common (int *arr1, int *arr2, int l1, int l2)

{

int count =0, flag1, i, j;

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

{

flag1=0;

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

{
```

```
if(arr1[i] == arr2[j])

{

flag1=1;  break;

}}

 if(flag1 ==0)

{

count++;

printf("%d,", arr1[i]);

}

}

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

{

flag1=0;

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

{
```

```
if(arr2[i] == arr1[j])

{

flag1=1;

break;

}}

if(flag1 ==0)

{

count++;

printf("%d,", arr2[i]);

}}

return count;

}

int main()

{

int len1=3,len2=3, result, i, j;

int arr1[10],arr2[10];
```

```
scanf("%d %d", &len1, &len2);

for(i=0; i<len1; i++)

scanf("%d", &arr1[i]);

for(i=0; i<len2; i++)

scanf("%d", &arr2[i]);

result = Not_common (arr1,arr2,len1,len2);

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

return 0;

}
```

Output: {1,3,4,5,6,8,10} Total is 7

Find the occurrence of a substring in a parent string

Input:

aAbcDefabcAdf

Substring : abc

In C

```
#include <stdio.h>

#include <string.h>

char str[100], sub[100];

int count = 0, count1 = 0;

int main()

{

int i, j, l, len1, len2;

printf("\nEnter a string : ");

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

len1 = strlen(str);

printf("\nEnter a substring : ");

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

len2 = strlen(sub);

for (i = 0; i < len1;)

{
```

```
j = 0,  count = 0;

while ((str[i] == sub[j]))

{

count++;   i++;  j++;

}

if (count == len2)

{          count1++;

count = 0;

}

}

printf("%d", count1);

return 0; }
```

Find the number of all possible triplets in the array that can form the triangle(condition is a+b>c) .

In C

```
#include <stdio.h>
int a[100];
int n,n1,n2;
int i,j,k;
int p,q,r;

int main(void)
{
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }

   for(i=0;i<n-2;i++)
   {
      p=a[i];
      for(j=i+1;j<n-1;j++)
      {
      q=a[j];
     for(k=j+1;k<n;k++)
         {
         r=a[k];
         if( ((p+q)>r) && ((p+r)>q) && ((q+r)>p) )
         {
         printf("%d  %d  %d  ",p,q,r);
         printf("Yes\n");
         }
         else
         {
         printf("%d  %d  %d  ",p,q,r);
```

```
            printf("No\n");
            }
            }
        }
    }
    return 0;
}
```

Print all the prime numbers which are below the given number separated by comma

Input: 50

In C

```
#include<stdio.h>

int main()

{

    int n,i,j,ct=0;

    scanf("%d",&n);

    for(i=2;i<=n;i++)

    {
```

```
        ct=0;
        for(j=2;j<i;j++)
        {
           if(i%j==0)
           {
             ct=1;
             break;
           }
        }
if(ct==0)
        {
           if(i>2)
           {
              printf(", ");
           }
           printf("%d",i);
```

```
        }
    }
}
```

Output: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47

Write a program to reverse a string

Input: PrepInsta

In C

```
#include <stdio.h>
#include <string.h>
void reverse(char [], int, int);
int main()
{
    char str1[100];
    int size;
```

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

    size = strlen(str1);

    reverse(str1, 0, size - 1);

    printf("%s", str1);

    return 0;

}

void reverse(char str1[], int index, int size)

{   char temp;

    temp = str1[index];

    str1[index] = str1[size - index];

    str1[size - index] = temp;

    if (index == size / 2)

    {

       return;

    }

    reverse(str1, index + 1, size);
```

```
}
```

Output: atsnIperP

Write a function to return a sorted array after merging two unsorted arrays, the parameters will be two integer pointers for referencing arrays and two int variable, the length of arrays (Hint: use malloc() to allocate memory for 3rd array):

In C

```
#include <stdio.h>

#include <stdlib.h>

int *mergesort(int *p,int *q, int n, int m)

{

   int i,index=0,j,t;

   static int arr3[20];

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

    {

       arr3[index]=*p;

       index++;
```

```
        p++;

    }

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

    {

        arr3[index]=*q;

        index++;

        q++;

    }

for(i = 1; i <= index; i++)

for(j = 1; j <= index; j++)

if(arr3[j-1] >= arr3[j])

{

  t = arr3[j-1];

arr3[j-1] = arr3[j];

arr3[j] = t;

}
```

```
        return arr3;
} int main()
{
  int arr1 [10], arr2 [10], *ptr;
     int i, n1, n2, m, index=0;
     scanf("%d", &n1);
     for(i=0;i<n1;i++)
     {
       scanf ("%d",&arr1[i]);
     }
      scanf ("%d", &n2 );
     for(i=0;i<n2;i++)
     {
       scanf ("%d", &arr2[i]);
       m = n1+n2;
```

```
    }

ptr=(int *) malloc( m * sizeof(int) );

ptr=mergesort(arr1, arr2, n1,n2);

for(i=1;i<=m;i++)

printf("%d\n", *(ptr+i));

return 0;    }
```

Write a program to return a sorted array from two unsorted array?

Examples:

Input : a[] = {10, 5, 15}
b[] = {20, 3, 2}
Output : Merge List :
{2, 3, 5, 10, 15, 20}

Input : a[] = {1, 10, 5, 15}
b[] = {20, 0, 2}
Output : Merge List :
{0, 1, 2, 5, 10, 15, 20}

Please also comment the code down below in other languages.

```
#include <bits/stdc++.h>
using namespace std;
```

```
// Function to merge array in sorted order
void sortedMerge(int a[], int b[], int res[],
int n, int m)
{
// Concatenate two arrays
int i = 0, j = 0, k = 0;
while (i < n) {
res[k] = a[i];
i += 1;
k += 1;
}
while (j < m) {
res[k] = b[j];
j += 1;
k += 1;
}

// sorting the res array
sort(res, res + n + m);
}

// Driver code
int main()
{
int a[] = { 10, 5, 15 };
int b[] = { 20, 3, 2, 12 };
int n = sizeof(a) / sizeof(a[0]);
int m = sizeof(b) / sizeof(b[0]);

// Final merge list
```

```
int res[n + m];
sortedMerge(a, b, res, n, m);

cout << “Sorted merged list :”;
for (int i = 0; i < n + m; i++)
cout << ” ” << res[i];
cout << “n”;

return 0;
}
```

program to print pattern

1

2*3

4*5*6

7*8*9*10

7*8*9*10

4*5*6

2*3

1

C

```
#include <stdio.h>

int main(void) {
	// your code goes here
```

```
    int n,i,j,k=1,temp;
    printf("Enter number:");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=i;j++)
        {
            printf("%d",k);
            if(i != j) printf("*");
            k++;
        }
        printf("\n");
    }
    k-=n;
    temp=k;
    for(i=n;i>=1;i--)
    {
        for(j=1;j<=i;j++)
        {
            printf("%d",temp);
            if(j == i) continue;
            else printf("*");
            temp++;
        }
        printf("\n");
        k-=(i-1);
        temp=k;
    }
return 0;
}
```

C++

```
#include <stdio.h>

#include <iostream>

using namespace std;

int main()

{

int n;

cin>>n;

int a=0;

for(int i=1;i<=n;i++)

{

for(int j=1;j<=i;j++)

{

a=a+1;

cout<<a;
```

```
if(j!=i)

cout<<"*";

}

cout<<"\n";

}

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

{

a=a-i;

for(int j=1;j<=i;j++)

{

cout<<a+j;

if(j!=i)

cout<<"*";

}

cout<<"\n";

}
```

```
return 0;

}
```

C

```
#include<stdio.h>

void pattern(int rows)

{

int i, j, n = 0;

for(i = 1; i <= rows; i++)

{

n++;

printf("%d", n);

for(j = 1; j <= i - 1; j++)

{

n++;

printf("*%d", n);
```

```
}

printf("\n");

}

for(i = rows; i >= 1; i--)

{

n = n - i + 1;

printf("%d", n);

for(j = 1; j <= i - 1; j++)

{

n++;

printf("*%d", n);

}

printf("\n");

n = n - i;

}
```

```
}

int main()

{

unsigned int rows;

printf("Enter number of rows(for upper or lower triangle): ");

scanf("%u", &rows);

pattern(rows);

}
```

Print the format in C programming

1*2*3*4

9*10*11*12

13*14*15*16

5*6*7*8

if you have to print the N number of rows where 1<=N<=100?

C

```
#include<stdio.h>

#include<stdlib.h>

int main(void)
```

```
{       int n,d,i;
            scanf("%d",&n);
                for(i=0;i<n;i+=2)
                    {       d=n*i+1;
                        for(int j=0;j<n;j++)
                            {       if(j==n-1)
                                        printf("%d",d);
                                    else    printf("%d*",d);
                                        d++;




                                    }
                        printf("\n");
}
                if(n%2)
                    i=n-2;
                else
                    i=n-1;



                    for(;i>0;i-=2)
                    {       d=n*i+1;
                        for(int j=0;j<n;j++)
                            {if(j==n-1)
                                        printf("%d",d);
                                    else    printf("%d*",d);
                                        d++;
```

```
                                                    }
                                                printf("\n");

}




return 0;}



C

---

void print(int n)
        {
           int m = 1;
           int i,j;
           for( i=1; i<=n-1; i++)
           {
              if(i!=2)
              {
                  for(j=1; j<=n; j++){
                              printf(“%d ”, m);
                              m++;
                  }
                  printf( "\n");
              }else{
                  m = m+n;
                  for(j=1; j<=n; j++){
                              printf(“%d ”, m);
                              m++;
```

```
                }
                printf("\n");
            }
        }
        for(i=n+1 ;i<=(n+n);i++){
                printf("%d ", i);
        }
        printf("\n");
    }
```

C++

```
#include <iostream>
using namespace std;
int main()
{
   int n;
   cout<<"Enter the Number of Rows : ";
   cin>>n;
   int p=n;
   if(n>=1 && n<=100)
   {
   for(int i=1;i<=n;i+=2)
   {
      int k=(i-1)*n+1;
      for(int j=0;j<n-1;j++)
      {
         cout<<k<<" * ";
         k++;
      }
      cout<<k<<" ";
```

```
        cout<<endl;
    }
    if(n%2!=0)
    {
        p=n-1;
    }
    for(int i=p;i>0;i-=2)
    {
        int k=(i-1)*n+1;
        for(int j=0;j<n-1;j++)
        {
            cout<<k<<" * ";
            k++;
        }
        cout<<k<<" ";
        cout<<endl;
    }
    }
    else
    {
        cout<<"Enter a Valid Input(1-100)!";
    }
    return 0;
}
```

C

(imp)

```
include <stdio.h>
```

```
int main()

{

int arr[100],n,i,j,count = 1, num=0;

scanf("%d",&n);

for(i=1;i<=n;i++)

{

if(i%2!=0){

for(j=1;j<=n;j++)

{

printf("%d",count);

if(j!=n)

printf("*");

count++;

}

printf("\n");

}
```

```
else{

for(j=1;j<=n;j++)

{

arr[num]=count;

count++;

num++;

}

}

}

for(i=1;i<=n/2;i++)

{

int num2=0;

num2=num-(i*n);

for(j=1;j<=n;j++)

{
```

```
printf("%d",arr[num2++]);

if(j!=n)

printf("*");

}

printf("\n");

}

return 0;

}
```

Pseudocode

```
for(i=0; i<N; i+=2)
        printf("%d*%d*%d*%d\n", (4*i+1), (4*i+2), (4*i+3), (4*i+4));
if(N%2)
        i-=2;
for(i--; i>0; i-=2)
        printf("%d*%d*%d*%d\n", (4*i+1), (4*i+2), (4*i+3), (4*i+4));
```

C

```
void main()
```

```
{

int i,n;

printf(“enter the rows:”);

scanf(“%d”,&n);

for(i=1; i<=n; i+=2)

printf(“%d*%d*%d*%d*\n”,(4*i+1),(4*i+2),(4*i+3),(4*i+4));

if(n%2)

i-=2;

for(i- -;i>n; i-=2)

printf(“%d*%d*%d*%d*\n”,(4*i+1),(4*i+2),(4*i+3),(4*i+4));

}
```

C++

```
#include <iostream>

using namespace std;

//Compiler version g++ 6.3.0
```

```
int main()

{

int n = 5;

int a1 = 1;

int a2 = 2*n + 1;

for(int i = 1; i <= n;i++){

for(int j = 1; j<= 2*n-1; j++){

if(j % 2 == 0){

cout<<"*";

}

else{

if(i==1 || i == n) {

cout<<a1;

a1++;

}
```

```
else{

cout<<a2;

a2++;

}

}

}

cout<<"\n";

}

}
```

C

```
#include<stdio.h>

int main(){

int m ;

int k = 0;

int i,j, n ;
```

```
printf("enter the number");

scanf("%d",&m);

if(m%2==0){

for(i=1 ;i<=(m/2);i++){

for(j=1;j<=m;j++){

printf("%d",(m*k)+j);

if(j<m){

printf("*");

}

}

printf("\n");

k = k+2;

}

int p , s;

int l = m-1;

for(p =1;p<=((m)/2);p++){
```

```
for(s=1;s<=m;s++){

printf("%d",(m*l)+s);

if(s<m){

printf("*");

}

}

printf("\n");

l = l-2;

}

}

else{

int g = m+1 ;

for(i=1 ;i<=(g/2);i++){

for(j=1;j<=m;j++){

printf("%d",(m*k)+j);
```

```
if(j<m){

printf("*");

}

}

printf("\n");

k = k+2;

}

int p , s;

int l = m-2;

for(p =1;p<((g)/2);p++){

for(s=1;s<=m;s++){

printf("%d",(m*l)+s);

if(s<m){

printf("*");

}

}
```

```
printf("\n");

l = l-2;

}

}

return 0;

}
```

C

```
public class Pattern{

public static void main(String[] args){

int n=13;

int p=n;

if(n>=1 && n<=100)

{

for(int i=1;i<=n;i+=2)

{ int k=(i-1)*n+1;
```

```
for(int j=0;j<n-1;j++)

{ System.out.print(k+"*");

k++;

}

System.out.print(k+" ");

System.out.println();

}

if(n%2!=0) p=n-1;

for(int i=p;i>0;i-=2)

{

int k=(i-1)*n+1;

for(int j=0;j<n-1;j++)

{

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

k++;
```

```
}

System.out.print(k+" ");

System.out.println();

}

}

}

}
```

C

```
#include <stdio.h>

int main(int n){

int i,j,N=n=4,k=1;

for(i=1;i<=N-1;i++)

{

if(i!=2){

for(j=1;j<=N;j++){

if(j==N){
```

```
printf("%d",k);

k++;}

else{

printf("%d*",k);

k++;}

}printf("\n");

}

else{

k=k+N;

for(j=1;j<=N;j++){

if(j==N){

printf("%d",k);

k++;

}

else{
```

```
printf("%d*",k);

k++;}

}

printf("\n");

}

}

/*for last line */

k=N+1;

for(j=1;j<=N;j++){

if(j==N){

printf("%d",k);

}

else{

printf("%d*",k);

k++;

}
```

```
}

return 0;

}
```

Basic Problem Solving:

You need to find the logic of the program which was given.

Find efficiency of the logic

logic optimization

```
int i=1;
for(int j =0;j<100;j++) //1<=N<=100
{
for(int k=0;k<4;k++)
{
printf("%d*",i++); //printing i variable on each loop
}
printf("\n"); // new line
}
```

JAVA

```
package a;
```

```
public class pattern {

public static void main(String...ss)

{

int n=6,j=1,k=n-1,p=0, l=n,count=1;

int[] a=new int[n];

if(n%2==0)

{

for(int i=0;i<=(n/2)-1;i++)

{

a[i]=j;

j=j+2;

}

for(int i=(n/2);i<n;i++)

{

a[i]=l;
```

```
l=l-2;

}

}

else

{

for(int i=0;i<=(n-1)/2;i++)

{

a[i] = j;

j = j+2;

}

for(int i=(n+1)/2;i<n;i++)

{

a[i] = k;

k=k-2;

}

}
```

```
int[][] b = new int[n][n];

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

{

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

{

b[i][t]=count;

count++;

}

}

while(p!=n)

{

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

{

System.out.printf("%02d",b[a[p]-1][i]);

if(i<n-1)
```

```
{

System.out.print('*');

}

}

System.out.println();

p++;

}

}

}
```

JAVA

```
import java.util.Scanner;

public class Pattern {

public static void main(String[] args)

{

Scanner sc=new Scanner(system.in -&nbspThis website is for sale! -&nbspSystem Resources and Information.);
```

```
System.out.println("Please enter the input");

int n=sc.nextInt();

int p=1;

int d=1;

if(n%2==0)

{

for(int i=1;i<=n;i++)

{

if(i==1)

{

for(int j=1;j<=2*n-1;j++)

{

if(j%2==1)

{

System.out.print(p++);
```

```
}

else

System.out.print("*");

}

System.out.println();

}

//*****************************end of first line*****************************//

if((i>1)&&(i<=n/2))

{

int j=1;

while(j<=2*n-1)/*for(int j=1;j<=2*n-1;j++)*/

{

if(j%2==1)

{

if(j==1)

{
```

```
p=p+n;

}

System.out.print(p);

p++;

}

else

System.out.print("*");

j++;

}

System.out.println();

}

//*********************************before half*************************************//

if(i==n/2+1)

{

for(int j=1;j<=2*n-1;j++)
```

```
{
if(j%2==1)
{
System.out.print(p++);
}
else
System.out.print("*");
}
System.out.println();
}
//**************after half***********//
if((i<n)&&(i>n/2))
{
int j=1;
while(j<=2*n-1)//for(int j=1;j<=2*n-1;j++)
{
```

```
if(j%2==1)

{

if(j==1)

{

p=p-n+1;

//d++;

p=p-(n*2)-d;

}

System.out.print(p);

p++;

}

else

System.out.print("*");

j++;

}
```

```
System.out.println();

}

//**************************after half*******************************//

}

}

else{

for(int i=1;i<=n;i++)

{

if(i==1)

{

for(int j=1;j<=2*n-1;j++)

{

if(j%2==1)

{

System.out.print(p++);

}
```

```
else

System.out.print("*");

}

System.out.println();

}

//*****************************end of first line*****************************//

if((i>1)&&(i<=n/2+1))

{

int j=1;

while(j<=2*n-1)/*for(int j=1;j<=2*n-1;j++)*/

{

if(j%2==1)

{

if(j==1)

{
```

```
p=p+n;

}

System.out.print(p);

p++;

}

else

System.out.print("*");

j++;

}

System.out.println();

}

//*********************************half ends*************************************//

//**************after half************//

if((i<n)&&(i>n/2))

{

int j=1;
```

```
while(j<=2*n-1)//for(int j=1;j<=2*n-1;j++)

{

if(j%2==1)

{

if(j==1)

{

p=p-(2*n)-n;

}

System.out.print(p+n);

p++;

}

else

System.out.print("*");

j++;

}
```

```
System.out.println();

}

//**************************after half*******************************//

}

}

}

}
```

C

```
#include <iostream>

#include <cstdio>

using namespace std;

int main() {

int N, i,;

cin>>N;

for(i=0; i<N; i+=2){
```

```
for(int j=1;j<=N;j++)

{

printf("%d", (4*i+j));

if(j!=N)

cout<<"*";

}

cout<<"\n";

}

if(N%2)

i-=2;

for(i--; i>0; i-=2){

for(int j=1;j<=N;j++)

{

printf("%d", (4*i+j));

if(j!=N)

cout<<"*";
```

```
}

cout<<"\n";

}

return 0;

}
```

C++

```
#include <iostream>

using namespace std;

int main()

{

// cout<<"Hello World";

int n,w=0,v=0,temp;

cout<<"row=";

cin >>n;

for(int i=1;i<=n;i++){
```

```
for(int j=1;j<=n;j++){

w++;

if(i%2==0){

continue;

}

else

cout<<w;

cout<<" ";

}

if(i%2==0){

continue;

}

else

cout<<"\n";

}
```

```
v=w;

for(int i=n;i>=1;i--){

v=v-n;

for(int j=1;j<=n;j++){

if(i%2==0){

cout<<v+j;

cout<<" ";

}

}

if(i%2==0){

cout<<"\n";

}

}

return 0;

}
```

C

```
void series() {

Scanner sc=new Scanner(system.in );

int m=sc.nextInt();

int n=1;

int count=1;

for(int i=1;i<m;i++) {

for(int j=1; j<=m;j++) {

System.out.print(n);

n++;

if(j<m) {

System.out.print("*");

}

}

count++;
```

```
n=(count*m)+1;

System.out.println();

}

for (int i=m+1; i<=m+m;i++) {

System.out.print(i);

if(i<m+m) {

System.out.print("*");

}

}

}
```

Print pattern in C using loops

1 2 3

6 5 4

7 8 9

12 11 10

C

```
#include <stdio.h>

int main()

{

int i,j,n;

printf(“Enter the number: ”);

scanf(“%d”,&n);

for(i=1;i<=n;i++)

{

printf(“%d ”,i);

if(i%3 == 0 && i != n)

{

j = i + 3;

printf(“\n”);

while(j != i)

{

if(j > n)
```

```
{

j--;

}

else

{

printf(“%d ”,j);

j--;

}

}

i = i + 3;

printf(“\n”);

}

}

return 0;

}
```

C++

```
#include <iostream>
using namespace std;
int main()
{
   int n,k;
   cout<<"Enter the Number of Rows : ";
   cin>>n;
   for(int i=0;i<n;i++)
   {
      if(i%2==0)
      {
         k=i*n+1;
         for(int j=0;j<n;j++)
         {
            cout<<k<<"  ";
            k++;
         }
      }
      else
      {
         k=(i+1)*n;
         for(int j=0;j<n;j++)
         {
            cout<<k<<"  ";
            k--;
         }
      }
      cout<<endl;
   }
```

```
   return 0;
}
```

```
bool backwd = false;
int count = 1;
int max = 100;
while(true) {
        if (count > max) {
                break;
        }
        if (backwd) {
                printf("%d, %d, %d\n", count + 2, count + 1, count);
        }
        else {
                printf("%d, %d, %d\n", count, count + 1, count + 2);
        }
        count += 3;
        backwd = !backwd;
}
```

C

--

```
#include<stdio.h>

/* Change this, make it an arg, etc. */
const int max = 24;

/* If you want 4 cols, make this 4, etc. */
const int cols = 3;
```

```
int main(int argc, char **argv) {
   int i = 0;
   int rev = 0;
   int val = 0;
   char sep = '\0';
   for (i = 0; i < max; ++i) {
      sep = (i + 1) % cols ? ' ' : '\n';
      rev = (i / cols) % 2;
      val = rev ? ((i / cols) + 1) * cols - (i % cols) : i + 1;
      printf("%d%c", val, sep);
   }
   return 0;
}
```

COCUBES CODING QUESTION – SUM OF INDIVIDUAL ROWS AND COLUMNS

FIND SUM LEAVING OUR ROW AND COL: COCUBES CODING QUESTION

PROBLEM STATEMENT:

You are given a function,

int FindSumLeavingOutRowCol(int** arr, int m, int n, int i,int j);

The function takes a two-dimensional array ‘arr’, its number of rows ‘m’, its number of columns ‘n’ and integers ‘i’ and ‘j’ as input. Implement the function to find and return the sum of elements of the array leaving out the elements of the i and j column. The algorithm to find the sum is as follows:

1.Iterate over every row except for i row,and keep on adding each element except for the elements of j column to a variable ‘sum’.

NOTE:

1.Row and column indices start from 0.

2. Value of i and j from 1.

CODING:

int FindSumLeavingOutRowCol(int** arr, int m, int n,int j);

```
{
/* write your code here */
}
**********
Program
**********
/*
 * C program to accept a matrix of order M x N and find the sum
 * of each row and each column of a matrix
 */
#include <stdio.h>

void main ()
{
   static int array[10][10];
   int i, j, m, n, sum = 0;

   printf(“Enter the order of the matrixn”);
   scanf(“%d %d”, &m, &n);
   printf(“Enter the co-efficients of the matrixn”);
   for (i = 0; i < m; ++i)
   {
      for (j = 0; j < n; ++j)
      {
         scanf(“%d”, &array[i][j]);
      }
   }
   for (i = 0; i < m; ++i)
   {
      for (j = 0; j < n; ++j)
      {
```

```
        sum = sum + array[i][j] ;
      }
      printf("Sum of the %d row is = %dn", i, sum);
      sum = 0;
   }
   sum = 0;
   for (j = 0; j < n; ++j)
   {
      for (i = 0; i < m; ++i)
      {
         sum = sum + array[i][j];
      }
      printf("Sum of the %d column is = %dn", j, sum);
      sum = 0;
   }
}
```

Write a program to print all the LEADERS in the array. An element is leader if it is greater than all the elements to its right side.

And the rightmost element is always a leader. For example int the array {16, 19, 4, 3, 8, 3}, leaders are 19, 8 and 3?

With C++

```
#include<iostream>
using namespace std;
```

```
/*C++ Function to print leaders in an array */
void printLeaders(int arr[], int size)
{
   for (int i = 0; i < size; i++)
   {
      int j;
      for (j = i+1; j < size; j++)
      {
         if (arr[i] <= arr[j])
            break;
      }
      if (j == size) // the loop didn't break
         cout << arr[i] << " ";
  }
}

/* Driver program to test above function */
int main()
{
   int arr[] = {16, 17, 4, 3, 5, 2};
   int n = sizeof(arr)/sizeof(arr[0]);
   printLeaders(arr, n);
   return 0;
}
```

With Java

```
class LeadersInArray
```

```
{

/*Java Function to print leaders in an array */

void printLeaders(int arr[], int size)

{

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

{

int j;

for (j = i + 1; j < size; j++)

{

if (arr[i] <= arr[j])

break;

}

if (j == size)

// the loop didn’t break System.out.print(arr[i] + ” “);

}

}
```

```
/* Driver program to test above functions */

public static void main(String[] args)

{

LeadersInArray lead = new LeadersInArray();

int arr[] = new int[]{16, 17, 4, 3, 5, 2};

int n = arr.length;

lead.printLeaders(arr, n);

}

}?
```

Maximum difference between two elements such that larger element appears after the smaller number

Given an array arr[] of integers, find out the difference between any two elements such that larger element appears after the smaller number in arr[].

Examples: If array is [2, 3, 10, 6, 4, 8, 1] then returned value should be 8 (Diff between 10 and 2). If array is [7, 9, 5, 6, 3, 2] then returned value should be 2 (Diff between 7 and 9)

Time Complexity: O(n^2)

Auxiliary Space: O(1)

Use two loops. In the outer loop, pick elements one by one and in the inner loop calculate the difference of the picked element with every other element in the array and compare the difference with the maximum difference calculated so far.

C

Java

C

```
#include

/* The function assumes that there are at least two
elements in array.
The function returns a negative value if the array is
sorted in decreasing order.
Returns 0 if elements are equal */
int maxDiff(int arr[], int arr_size)
{
int max_diff = arr[1] – arr[0];
int i, j;
for (i = 0; i < arr_size; i++)
{
for (j = i+1; j < arr_size; j++)
{
if (arr[j] – arr[i] > max_diff)
max_diff = arr[j] – arr[i];
}
}
return max_diff;
}

/* Driver program to test above function */
int main()
```

```
{
int arr[] = {1, 2, 90, 10, 110};
printf("Maximum difference is %d", maxDiff(arr, 5));
getchar();
return 0;
}
```

Java

```
class MaximumDiffrence
{
/* The function assumes that there are at least two
elements in array.
The function returns a negative value if the array is
sorted in decreasing order.
Returns 0 if elements are equal */
int maxDiff(int arr[], int arr_size)
{
int max_diff = arr[1] – arr[0];
int i, j;
for (i = 0; i < arr_size; i++)
{
for (j = i + 1; j < arr_size; j++)
{
if (arr[j] – arr[i] > max_diff)
max_diff = arr[j] – arr[i];
}
}
return max_diff;
}
```

```
/* Driver program to test above functions */
public static void main(String[] args)
{
MaximumDiffrence maxdif = new MaximumDiffrence();
int arr[] = {1, 2, 90, 10, 110};
System.out.println("Maximum differnce is " +
maxdif.maxDiff(arr, 5));
}
```

Tricky Solution but Efficient

Method 2 (Tricky and Efficient)

In this method, instead of taking difference of the picked element with every other element, we take the difference with the minimum element found so far. So we need to keep track of 2 things:

1) Maximum difference found so far (max_diff).

2) Minimum number visited so far (min_element).

Time Complexity: O(n)

Auxiliary Space: O(1)

C

```
#include<stdio.h>

/* The function assumes that there are at least two
elements in array.
The function returns a negative value if the array is
sorted in decreasing order.
Returns 0 if elements are equal */
int maxDiff(int arr[], int arr_size)
{
int max_diff = arr[1] – arr[0];
```

```
int min_element = arr[0];
int i;
for(i = 1; i < arr_size; i++)
{
if (arr[i] – min_element > max_diff)
max_diff = arr[i] – min_element;
if (arr[i] < min_element)
min_element = arr[i];
}
return max_diff;
}

/* Driver program to test above function */
int main()
{
int arr[] = {1, 2, 6, 80, 100};
int size = sizeof(arr)/sizeof(arr[0]);
printf(“Maximum difference is %d”, maxDiff(arr, size));
getchar();
return 0;
```

Java

```
class MaximumDiffrence
{
/* The function assumes that there are at least two
elements in array.
The function returns a negative value if the array is
sorted in decreasing order.
Returns 0 if elements are equal */
int maxDiff(int arr[], int arr_size)
```

```
{
int max_diff = arr[1] – arr[0];
int min_element = arr[0];
int i;
for (i = 1; i < arr_size; i++)
{
if (arr[i] – min_element > max_diff)
max_diff = arr[i] – min_element;
if (arr[i] < min_element)
min_element = arr[i];
}
return max_diff;
}

/* Driver program to test above functions */
public static void main(String[] args)
{
MaximumDiffrence maxdif = new MaximumDiffrence();
int arr[] = {1, 2, 90, 10, 110};
int size = arr.length;
System.out.println("MaximumDifference is " +
maxdif.maxDiff(arr, size));
}
}
```

Given a string of character, find the length of longest proper prefix which is also a proper suffix.

Example:

S = abab

lps is 2 because, ab.. is prefix and ..ab is also a suffix.

Input:

First line is T number of test cases. 1<=T<=100.

Each test case has one line denoting the string of length less than 100000.

Expected time compexity is O(N).

Output:

Print length of longest proper prefix which is also a proper suffix.

Example:

Input:

2

abab

aaaa

Output:

2

3

C++

```
#include < bits / stdc++.h >
using namespace std;
int lps(string);
int main() {
//code
int T;
cin >> T;
getchar();
while (T–) {
string s;
```

```
cin >> s;
printf(“%d\n”, lps(s));
}
return 0;
}
int lps(string s) {
int n = s.size();
int lps[n];
int i = 1, j = 0;
lps[0] = 0;
while (i < n) {
if (s[i] == s[j]) {
j++;
lps[i] = j;
i++;
} else {
if (j != 0)
j = lps[j – 1];
else {
lps[i] = 0;
i++;
}
}
}
return lps[n – 1];
}
```

Java

```
import java.util.*;
import java.lang.*;
```

```
import java.io.*;
class PreSuf {
public static void main(String[] args) {
//code
Scanner s = new Scanner(System.in);
int t = s.nextInt();
for (int i = 0; i < t; i++) {
String s1 = s.next();
int j = 1, k = 0, l = s1.length(), max = 0, len = 0;
int lps[] = new int[l];
while (j < l) {
if (s1.charAt(j) == s1.charAt(len)) {
len++;
lps[j] = len;
j++;
} else {
if (len != 0) {
len = lps[len – 1];
} else {
lps[j] = 0;
j++;
}
}
}

System.out.println(lps[l – 1]);
}
}
}
```

Find the number closest to n and divisible by m

Given two integers n and m. The problem is to find the number closest to n and divisible by m. If there are more than one such number, then output the one having maximum absolute value. If n is completely divisible by m, then output n only. Time complexity of O(1) is required.

Constraints: m != 0

We find value of n/m. Let this value be q. Then we find closest of two possibilities. One is q * m other is (m * (q + 1)) or (m * (q – 1)) depending on whether one of the given two numbers is negative or not.

Algorithm:

```
closestNumber(n, m)
   Declare q, n1, n2
   q = n / m
   n1 = m * q

   if (n * m) > 0
      n2 = m * (q + 1)
   else
      n2 = m * (q - 1)

   if abs(n-n1) < abs(n-n2)
      return n1
   return n2
```

C++

```
// C++ implementation to find the number closest to n
// and divisible by m
#include <bits/stdc++.h>
```

```
using namespace std;

// function to find the number closest to n
// and divisible by m
int closestNumber(int n, int m)
{
// find the quotient
int q = n / m;

// 1st possible closest number
int n1 = m * q;

// 2nd possible closest number
int n2 = (n * m) > 0 ? (m * (q + 1)) : (m * (q – 1));

// if true, then n1 is the required closest number
if (abs(n – n1) < abs(n – n2))
return n1;

// else n2 is the required closest number
return n2;
}

// Driver program to test above
int main()
{
int n = 13, m = 4;
cout << closestNumber(n, m) << endl;

n = -15; m = 6;
cout << closestNumber(n, m) << endl;
```

```
n = 0; m = 8;
cout << closestNumber(n, m) << endl;

n = 18; m = -7;
cout << closestNumber(n, m) << endl;

return 0;
}
```

Java

```
// Java implementation to find the number closest to n
// and divisible by m
public class close_to_n_divisible_m {

// function to find the number closest to n
// and divisible by m
static int closestNumber(int n, int m)
{
// find the quotient
int q = n / m;

// 1st possible closest number
int n1 = m * q;

// 2nd possible closest number
int n2 = (n * m) > 0 ? (m * (q + 1)) : (m * (q – 1));

// if true, then n1 is the required closest number
if (Math.abs(n – n1) < Math.abs(n – n2))
return n1;
```

```
// else n2 is the required closest number
return n2;
}

// Driver program to test above
public static void main(String args[])
{
int n = 13, m = 4;
System.out.println(closestNumber(n, m));

n = -15; m = 6;
System.out.println(closestNumber(n, m));

n = 0; m = 8;
System.out.println(closestNumber(n, m));

n = 18; m = -7;
System.out.println(closestNumber(n, m));
}
}
```

Given a string consisting of only 0, 1, A, B, C where

A = AND

B = OR

C = XOR

Calculate the value of the string assuming no order of precedence and evaluation is done from left to right.

Constraints – The length of string will be odd. It will always be a valid string.

Example, 1AA0 will not be given as an input.

Examples:

Input : 1A0B1

Output : 1

1 AND 0 OR 1 = 1

Input : 1C1B1B0A0

Output : 0

C/C++

```
// C++ program to evaluate value of an expression.
#include <bits/stdc++.h>
using namespace std;

int evaluateBoolExpr(string s)
{
int n = s.length();

// Traverse all operands by jumping
// a character after every iteration.
for (int i = 0; i < n; i += 2) {

// If operator next to current operand
// is AND.
if (s[i + 1] =='A') {
if (s[i + 2] =='0'|| s[i] =='0')
s[i + 2] ='0';
else
s[i + 2] ='1';
}
```

```
// If operator next to current operand
// is OR.
else if (s[i + 1] ==’B’) {
if (s[i + 2] ==’1’|| s[i] ==’1’)
s[i + 2] =’1’;
else
s[i + 2] =’0’;
}

// If operator next to current operand
// is XOR (Assuming a valid input)
else {
if (s[i + 2] == s[i])
s[i + 2] =’0’;
else
s[i + 2] =’1’
}
}
return s[n – 1] -‘0’;
}

// Driver code
int main()
{
string s = “1C1B1B0A0”;
cout << evaluateBoolExpr(s);
return 0;
}
```

Java

```
// Java program to evaluate value of an expression.
public class Evaluate_BoolExp {

// Evaluates boolean expression
// and returns the result
static int evaluateBoolExpr(StringBuffer s)
{
int n = s.length();

// Traverse all operands by jumping
// a character after every iteration.
for (int i = 0; i < n; i += 2) {

// If operator next to current operand
// is AND.
if( i + 1 < n && i + 2 < n)
{
if (s.charAt(i + 1) == 'A') {
if (s.charAt(i + 2) == '0' ||
s.charAt(i) == 0)
s.setCharAt(i + 2, '0');
else
s.setCharAt(i + 2, '1');
}

// If operator next to current operand
// is OR.
else if ((i + 1) < n &&
s.charAt(i + 1 ) == 'B') {
if (s.charAt(i + 2) == '1' ||
s.charAt(i) == '1')
```

```
s.setCharAt(i + 2, ‘1’);
else
s.setCharAt(i + 2, ‘0’);
}

// If operator next to current operand
// is XOR (Assuming a valid input)
else {
if (s.charAt(i + 2) == s.charAt(i))
s.setCharAt(i + 2, ‘0’);
else
s.setCharAt(i + 2 ,’1');
}
}
}
return s.charAt(n – 1) – ‘0’;
}

// Driver code
public static void main(String[] args)
{
String s = “1C1B1B0A0”;
StringBuffer sb = new StringBuffer(s);
System.out.println(evaluateBoolExpr(sb));
}
}



return s[n – 1] -‘0’;
}
```

```
// Driver code

int main()

{

string s = “1C1B1B0A0”;

cout << evaluateBoolExpr(s);

return 0;

}
```

Ques. 1 Write a program to find out total marks obtained by a student if the student gets 3 marks for the correct answer and -1 for the wrong answer?

You have to classify a string as “GOOD”, “BAD” or “MIXED”. A string is composed of lowercase alphabets and ‘?’. A ‘?’ is to be replaced by any of the lowercase alphabets. Now you have to classify the string on the basis of some rules. If there are more than 3 consonants together, the string is considered to be “BAD”. If there are more than 5 vowels together, the also string is considered to be “BAD”. A string is “GOOD” if its not “BAD”. Now when question marks are involved, they can be replaced with consonants or vowels to make new strings. If all the choices lead to “GOOD” strings then the input is considered as “GOOD”, and if all the choices lead to “BAD” strings then the input is “BAD”, else the string is “MIXED?

LOGIC

1. Convert the string in **0’s and 1’s**. All consonants will be considered as 1 and vowels as 0.
2. Make 2 D array (this is dynamic programming approach) and start matching.
3. For string matching, one side will be the string (converted with 0’s and 1’s) and another side with vowels and consonants as 0 and 1.

cons & Vow/ String		W	A	Y	T	O	C	R	A	C	K
	0	1	0	1	1	0	1	1	0	1	1
0	0	0	1	0	0	1	0	0	1	0	0
1	0	1	0	1	2	0	1	2	0	1	2

1. Whenever the 0 will be matched with 0, increment previous array element by 1. Similarly when 1 will be matching with 1, increment previous array element by 1.
2. In case of 3 consecutive consonants, the current array value reaches to 3 or 5 consecutive vowels the string is said to **“BAD**”.

Now here comes ? marks case

cons & Vow/ String		W	A	Y	T	**?**	C	R	A	C	K
	0	1	0	1	1	**?**	1	1	0	1	1
0	0	0	1	0	0	**1**	0	0	1	0	0
1	0	1	0	1	2	**3**	**4**	**5**	0	1	2

I**n case of question mark "?" add one to both consonant and vowel array**. Also, make a Boolean flag which will be true in case of "?" occurs. If "?" and 3 consecutive consonants or 5 consecutive vowels occurred then the string is said to be **"MIXED"**, otherwise the string is **"GOOD**".

Please see below code for clear understanding, some of the corner test cases are tested which are also commented there, to test uncomment it and run.

Some more test cases are also given at the end of this post.

Code –

Find if string is good, bad or mixed Code

```
#include "string.h"
#include"iostream"
using namespace std;

//alphabet check
bool alphabetcheck(char alphabet){
        if (alphabet >= 'a' && alphabet <= 'z')
            return true;
        return false;
    }

//vowel check
int vowelcheck(char vowel)
{
        if (vowel == 'a' || vowel == 'e' || vowel == 'i' || vowel == 'o' || vowel
== 'u')
            return 0;
        return 1;
}


int main(void) {
    //the two sequences
    //string X = "waytocrack";// ----  GOOD
    //string X = "waaaaaaytocrack";//--BAD
    string X = "wayt?arack";  // mixed
    //length of the sequences
    int XLen = X.size();
    int Arr[2][20];
    memset(Arr, 0, sizeof(Arr[0][0]) * 2 * 20);
```

```
    int max0 = 0;
    int max1 = 0;
    int index;
    bool mixed_value = false;
    for (size_t i = 0; i < XLen; i++)
    {
        //alphabet check
        if (alphabetcheck(X[i]))
        {
            //vowel check fucntion:
            if ((vowelcheck(X[i])) == 0)
            {
                Arr[0][i+1] = Arr[0][i] + 1;
                if (Arr[0][i + 1] > max0)
                    max0 = Arr[0][i + 1]; //check for max 0
                if ((mixed_value == true) && (max0 >= 5) && (Arr[0][i + 1] >= 5))
// Arr[0][i + 1] >= 5 when current value is greater than 5
                {
                    if ((i - index >= 5 || (Arr[1][index] + Arr[0][index + 5] ==
7)))
                     mixed_value = false;
                }
            }
            else
            {
                Arr[1][i + 1] = Arr[1][i] + 1;
                if (Arr[1][i + 1] > max1)
                    max1 = Arr[1][i + 1]; //check for max 1
                if (mixed_value == true && max1 >= 3 && Arr[1][i + 1] >= 3 )
 // Arr[0][i + 1] >= 5 when current value is greater than 3
                {
                    if ((i - index >= 3 || (Arr[0][index] + Arr[1][index + 3] ==
7)) )
                        mixed_value = false;
                }
            }
            if (mixed_value == false && (max0 >= 5 || max1 >= 3)){
//checking the count value GREATER than  or 3
                cout << " BAD " << endl;
                exit(0);
            }
        }

        else if (X[i] == '?'){
            //increament both o count and 1 count.
            Arr[0][i + 1] = Arr[0][i] + 1;
            if (Arr[0][i + 1] > max0)
                max0 = Arr[0][i + 1]; //check for max 1
            Arr[1][i + 1] = Arr[1][i] + 1;
            if (Arr[1][i + 1] > max1)
                max1 = Arr[1][i + 1]; //check for max 1
            index = i;
```

```
            mixed_value = true;
        }
    }


    if (mixed_value & (max0 >= 5 || max1 >= 3))
        cout << " MIXED " << endl;
    else cout << " GOOD " << endl;

    return 0;
}
```

These are some corner test cases:

a?fafff	BAD
??aa??	MIXED
abc	GOOD
aaa?aaafff	BAD
aaaa?ff?aaa?aaa?fff	BAD
aaaaff?	MIXED
aaaaf?	GOOD
?aaaaffaaf?aaaafff	BAD
?aaaaffaaf?aaaaff	MIXED
vaxaaaa?bbadadada	BAD
aaaa?bb	BAD
vabb?aaaadadada	BAD
vabab?aaaadadada	MIXED

The solution is 3 or more than 3 consecutive consonants and 5 and more than 5 consecutive vowels.
You can check some of the tested test cases at the end of the code.

We are given an array with n elements from {1,2,3,4}. Find the number of minimum changes required to be performed so that no two adjacent numbers are same?

We are given a count of songs to be played – n, highest volume allowed – h, initial volume – i, and list of allowed volume change A[] of size n. The singer can either increase/decrease the volume of sound system for the next song by the allowed volume change A[j] for jth song from the volume of the j-1th song. The aim is to maximize the volume of last sound. Find the maximum volume that can be attained, or return -1 if there is no possibility of changing volume due to the given constrains. (Volume cannot be in negative.)

Write a program to print all Subsequences of String which Start with Vowel and End with Consonant

Given a string return all possible subsequences which start with vowel and end with a consonant. A String is a subsequence of a given String, that is generated by deleting some character of a given string without changing its order.
Examples:

Input : ‘abc’
Output : ab, ac, abc

Input : ‘aab’
Output : ab, aab

xplanation of the Algorithm:

```
Step 1: Iterate over the entire String
Step 2: check if the ith character for vowel
Step 3: If true iterate the string from the end,
        if false move to next iteration
Step 4: check the jth character for consonent
        if false move to next iteration
        if true perform the following
Step 5: Add the substring starting at index i and
        ending at index j to the hastset.
Step 6: Iterate over the substring drop each character
        and recur to generate all its subString
```

```
// Java Program to generate all the subsequence
// starting with vowel and ending with consonant.
import java.util.HashSet;
public class Subsequence {
    // Set to store all the subsequences
    static HashSet<String> st = new HashSet<>();
    // It computes all the possible substring that
    // starts with vowel and end with consonent
    static void subsequence(String str)
    {
        // iterate over the entire string
        for (int i = 0; i < str.length(); i++) {

            // test ith character for vowel
            if (isVowel(str.charAt(i))) {

                // if the ith character is vowel
                // iterate from end of the string
                // and check for consonant.
```

```
                for (int j = (str.length() - 1); j >= i; j--) {

                    // test jth character for consonant.
                    if (isConsonant(str.charAt((j)))) {

                        // once we get a consonant add it to
                        // the hashset
                        String str_sub = str.substring(i, j + 1);
                        st.add(str_sub);
                        // drop each character of the substring
                        // and recur to generate all subsquence
                        // of the substring
                        for (int k = 1; k < str_sub.length() - 1; k++) {
                            StringBuffer sb = new StringBuffer(str_sub);
                            sb.deleteCharAt(k);
                            subsequence(sb.toString());
                        }
                    }
                }
            }
        }
    }
    // Utility method to check vowel
    static boolean isVowel(char c)
    {
        return (c == 'a' || c == 'e' || c == 'i' || c == 'o'
                                            || c == 'u');
    }
    // Utility method to check consonant
    static boolean isConsonant(char c)
    {
        return !(c == 'a' || c == 'e' || c == 'i' || c == 'o'
                                            || c == 'u');
    }
    // Driver code
    public static void main(String[] args)
    {
        String s = "xabcef";
        subsequence(s);
        System.out.println(st);
    }
```

```
}
```

Output:

```
[ef, ab, ac, aef, abc, abf, af, acf, abcef, abcf, acef, abef]
```

Ques. Find winner of an election where votes are represented as candidate namesGiven an array of names of candidates in an election. A candidate name in array represents a vote casted to the candidate. Print the name of candidates received Max vote. If there is tie, print lexicographically smaller name.A **simple solution** is to run two loops and count occurrences of every word. Time complexity of this solution is O(n * n * MAX_WORD_LEN).An **efficient solution** is to use Hashing. We insert all votes in a hash map and keep track of counts of different names. Finally we traverse the map and print the person with maximum votes.

```
// Java program to find winner in an election.
import java.util.*;
public class ElectoralVotingBallot
{
    /* We have four Candidates with name as 'John',
      'Johnny', 'jamie', 'jackie'.
       The votes in String array are as per the
       votes casted. Print the name of candidates
       received Max vote. */
    public static void findWinner(String votes[])
    {
        // Insert all votes in a hashmap
        Map<String,Integer> map =
                        new HashMap<String, Integer>();
        for (String str : votes)
        {
            if (map.keySet().contains(str))
                map.put(str, map.get(str) + 1);
            else
                map.put(str, 1);
        }
        // Traverse through map to find the candidate
        // with maximum votes.
        int maxValueInMap = 0;
        String winner = "";
```

```
        Map.Entry<String,Integer> entry;
        for (entry : map.entrySet())
        {
            String key  = entry.getKey();
            Integer val = entry.getValue();
            if (val > maxValueInMap)
            {
                maxValueInMap = val;
                winner = key;
            }
            // If there is a tie, pick lexicographically
            // smaller.
            else if (val == maxValueInMap &&
                winner.compareTo(key) > 0)
                winner = key;
        }
        System.out.println("Winning Candidate is :" +
                                                winner);
    }
    // Driver code
    public static void main(String[] args)
    {
        String[] votes = { "john", "johnny", "jackie",
                           "johnny", "john", "jackie",
                           "jamie", "jamie", "john",
                           "johnny", "jamie", "johnny",
                           "john" };
        findWinner(votes);
    }
}
```

Output:

```
John
```

A man starts from his house with a few pan cakes. let they be N. Now he visits K places before reaching home. At each place he can buy a cake, sell a cake or do nothing. But he must sell L cakes before reaching home.Find the maximum number of cakes he can have at any point in his journey.N,K,L are given as input?

A man starts from his house with a few pan cakes. let they be N. Now he visits K places before reaching home. At each place he can buy a cake, sell a cake or do nothing. But he must sell L cakes before reaching home.Find the maximum number of cakes he can have at any point in his journey.N,K,L are given as input?

Find the row with maximum number of 1s?

Given a boolean 2D array, where each row is sorted. Find the row with the maximum number of 1s.

```
Example
Input matrix
0 1 1 1
0 0 1 1
1 1 1 1  // this row has maximum 1s
0 0 0 0

Output: 2
```

A simple method is to do a row wise traversal of the matrix, count the number of 1s in each row and compare the count with max. Finally, return the index of row with maximum 1s. The time complexity of this method is O(m*n) where m is number of rows and n is number of columns in matrix.

We can do better. Since each row is sorted, we can **use Binary Search** to count of 1s in each row. We find the index of first instance of 1 in each row. The count of 1s will be equal to total number of columns minus the index of first 1.

See the following code for implementation of the above approach.

```
#include <stdio.h>
#define R 4
#define C 4
/* A function to find the index of first index of 1 in a boolean array arr[] */
int first(bool arr[], int low, int high)
{
  if(high >= low)
  {
    // get the middle index
    int mid = low + (high - low)/2;
    // check if the element at middle index is first 1
    if ( ( mid == 0 || arr[mid-1] == 0) && arr[mid] == 1)
      return mid;
    // if the element is 0, recur for right side
    else if (arr[mid] == 0)
```

```
      return first(arr, (mid + 1), high);
    else // If element is not first 1, recur for left side
      return first(arr, low, (mid -1));
  }
  return -1;
}
// The main function that returns index of row with maximum number of 1s.
int rowWithMax1s(bool mat[R][C])
{
    int max_row_index = 0, max = -1; // Initialize max values
    // Traverse for each row and count number of 1s by finding the index
    // of first 1
    int i, index;
    for (i = 0; i < R; i++)
    {
       index = first (mat[i], 0, C-1);
       if (index != -1 && C-index > max)
       {
           max = C - index;
           max_row_index = i;
       }
    }
    return max_row_index;
}
/* Driver program to test above functions */
int main()
{
    bool mat[R][C] = { {0, 0, 0, 1},
        {0, 1, 1, 1},
        {1, 1, 1, 1},
        {0, 0, 0, 0}
    };
    printf("Index of row with maximum 1s is %d n", rowWithMax1s(mat));
    return 0;
}
```

Output:

```
Index of row with maximum 1s is 2
```

Time Complexity: O(mLogn) where m is a number of rows and n is a number of columns in a matrix.

1. Program to check if a Binary tree is BST or note
2. Find the Lowest Common Ancestor in a Binary Search Tree
3. Program to check if a binary tree is height balanced or not
4. Program Two Nodes of a BST are Swapped correct the BST
5. Find maximum in binary tree
6. Program to Connect nodes of a Tree at same level
7. Find the Lowest Common Ancestor in a Binary Tree
8. Add Two Numbers Represented by Linked Lists
9. Detecting Loop in a Linked List