Arrays (One Dimension Questions)

Q1. Write a program to create an array, store values in it and display them.

```
#include<stdio.h>
void main()
{
int roll_no[10];
int index;
printf("\n INPUT THE VALUES BELOW :\n");
for(index=1;index<=10;index++)
{
printf("Enter the element at position %d is:",index);
scanf("%d",&roll_no[index]);
}
printf("\n INSERTED VALUES ARE :\n");
for(index=0;index<10;index++)
{
printf("\nthe element at %d index is : %d",index,roll_no[index]);
}
```

OR

```
#include<stdio.h>
void main()
{
int roll_no[5]={1915001,1915002,1915003,1915004,1915005};
printf("%d\n",roll_no[0]);
printf("%d\n",roll_no[1]);
printf("%d\n",roll_no[2]);
printf("%d\n",roll_no[3]);
printf("%d\n",roll_no[4]);
int index;
for(index=0;index<5;index++)
{
printf("%d\n",roll_no[index]);
}
}
```

OR

```
#include<stdio.h>
void main()
{
```

```
int roll_no[10]={1,2,3,4,5,6,7,8,9,10};
int index;
printf("\n INSERTED VALUES ARE :\n");
for(index=0;index<10;index++)
{
printf("\nthe largest element is %d",roll_no[index]);
}
```

OR

```
#include<stdio.h>
void main()
{
int roll_no[10];
roll_no[0]=1;
roll_no[1]=2;
roll_no[2]=3;
roll_no[3]=4;
roll_no[4]=5;
roll_no[5]=6;
roll_no[6]=7;
roll_no[7]=8;
roll_no[8]=9;
roll_no[9]=10;
int index;
printf("\n INSERTED VALUES ARE :\n");
for(index=0;index<10;index++)
{
printf("\nthe largest element is %d",roll_no[index]);
}
```

Q2. Write a program to create an array and display it in reverse order.

```
#include<stdio.h>
void main()
{
int roll_no[10];
int index;
printf("\n INPUT THE VALUES BELOW :\n");
for(index=1;index<=10;index++)
{
printf("Enter the element at position %d is:",index);
scanf("%d",&roll_no[index]);
}
```

```
printf("\n INSERTED VALUES ARE :\n");
for(index=9;index>=0;index–)
{
printf("\nthe element at %d index is : %d",index,roll_no[index]);
}
```

Q3. Write a program to display largest element of the array.

```
#include<stdio.h>
void main()
{
int arr[500],index,max,n;
printf("Enter the total number of elements you want to enter in the array:");
scanf("%d",&n);
for(index=0;index<n;index++)
{
printf("\nElement at position %d is :",index);
scanf("%d",&arr[index]);
}
max=arr[0];
for(index=0;index<n;index++)
{
if(arr[index]>max)
{
max=arr[index];
}
}
printf("The largest element in array is :%d",max);
}
```

OR

```
#include<stdio.h>
void main()
{
int arr[500],index,n;
printf("Enter the total number of elements you want to enter in the array:");
scanf("%d",&n);
for(index=0;index<n;index++)
{
printf("\nElement at position %d is :",index);
scanf("%d",&arr[index]);
}
for(index=0;index<n;index++)
{
```

```
if(arr[index]>arr[0])
{
arr[0]=arr[index];
}
}
printf("The largest element in array is :%d",arr[0]);
}
```

Q4. Write a program to display the smallest element of an array.

```
#include<stdio.h>
void main()
{
int arr[500],index,min,n;
printf("Enter the total number of elements you want to enter in the array:");
scanf("%d",&n);
for(index=0;index<n;index++)
{
printf("\nElement at position %d is :",index);
scanf("%d",&arr[index]);
}
min=arr[0];
for(index=0;index<n;index++)
{
if(arr[index]<min)
{
min=arr[index];
}
}
printf("The smallest element in array is :%d",min);
}
```

Q5. Write a program to display the sum of elements of an array.

```
#include<stdio.h>
void main()
{
int data[10],index;
printf("Enter the elements in array:");
for(index=0;index<10;index++)
{
scanf("%d",&data[index]);
}
printf("The sum of entered values are:");
for(index=0;index<10;index++)
```

```
{
sum=sum+data[index];
}
printf("%d",sum);
}
```

Q6. Write a program to copy the elements of an array to another array.

```
void main()
{
int ori[10],dup[10],index;
printf("Enter the elements in original array:");
for(index=0;index<10;index++)
{
printf("\nEnter the elements at %d index:",index);
scanf("%d",&ori[index]);
}
printf("\nThe copy the original data to duplicate array:");
for(index=0;index<10;index++)
{
dup[index]=ori[index];
}
printf("\n After copy the duplicate array is:\n");
for(index=0;index<10;index++)
{
printf("\nThe elements at %d index:",index);
printf("%d",dup[index]);
}
}
```

Q7. Write a program to insert an element in an array.

```
#include<stdio.h>
void main()
{
int arr[500],index,pos,n,new;
printf("Enter the total number of elements you want to enter in the array:");
scanf("%d",&n);
for(index=0;index<n;index++)
{
printf("\nElement at position %d is :",index);
scanf("%d",&arr[index]);
}
printf("\nEnter the position at which you want to insert the new element:");
```

```
scanf(“%d”,&pos);
printf(“\nEnter the new element:”);
scanf(“%d”,&new);
for(index=n;index>=pos;index–)
{
arr[index]=arr[index-1];
}
arr[pos-1]=new;
printf(“\n array after inserting new element:\n”);
for(index=0;index<=n;index++)
{
printf(“\nthe element at position %d is : %d”,index,arr[index]);
}
}
```

Q8. Write a program to delete an element from an array.

```
#include <stdio.h>
#define MAX_SIZE 500
void main()
{
int arr[MAX_SIZE];
int index, size, pos;
/* input size and element in array */
printf(“Enter size of the array : “);
scanf(“%d”, &size);//5
printf(“Enter elements in array : “);
for(index=0; index<size; index++)
{
scanf(“%d”, &arr[index]);
}
/* input element position to delete */
printf(“Enter the element position to delete : “);
scanf(“%d”, &pos);
/* invalid delete position */
if(pos < 0 || pos > size)
{
printf(“invalid position! Please enter position between 1 to %d”, size);
}
else
{
/* Copy next element value to current element */
for(index=pos-1; index<size-1; index++)
{
```

```
arr[index] = arr[index + 1];
}
/* Decrement array size by 1 */
size–;
}
/* Print array after deletion */
printf(“\nElements of array after delete are : “);
for(index=0; index<size; index++)
{
printf(“%d\t”, arr[index]);
}
}
```

Q9. Write a program to perform Linear search in an array.

```
#include<stdio.h>
void main()
{
int data[100],size,index,search,found=0;
scanf(“%d”,&size);5
for(index=0;index<size;index++)
{
scanf(“%d”,&data[index]);
}
printf(“\nEnter the element you want to search:”);
scanf(“%d”,&search);
for(index=0;index<size;index++)
{
if(data[index]==search)
{
found++;
break;
}
}
if(found==1)
{
printf(“\n Element %d is found at %d index”,search,index);
}
else
{
printf(“Element not ava”);
}
}
```

OR

```
#include<stdio.h>
void main()
{
int data[100],size,index,search;
printf("Enter the size of array:");
scanf("%d",&size);//5
printf("\nEnter the elements in array:\n");
for(index=0;index<size;index++)
{
scanf("%d",&data[index]);//5 78 46 89 3
}
printf("\nEnter the element you want to search:");
scanf("%d",&search);//40
for(index=0;index<size;index++)
{
if(data[index]==search)
{
printf("\nElement is found at %d index",index);
}
}
```

Q 10. Write a program to perform bubble shorting in ascending order.
Input :- 10 22 8 7 15 21
Output :- 7 8 10 15 21 22*/

```
#include<stdio.h>
void main()
{
int data[1000],size,index,j,temp;
scanf("%d",&size);//6
for(index=0;index<size;index++)
{
scanf("%d",&data[index]);//7 8 10 15 21 22
}
for(index=0;index<size;index++)
{
for(j=index+1;j<size;j++)
{
if(data[index]>data[j])
{
temp=data[index];
data[index]=data[j];
data[j]=temp;
```

```
}
}
}
for(index=0;index<size;index++)
{
printf("\n%d",data[index]);
}
}
```

Q11. A program P reads in 500 integers in the range [0..100] representing the scores of 500 students. It then prints the frequency of each score above 50. What would be the best way for P to store the frequencies? Write the above program in 'C'.

```
#include"stdio.h"
void main()
{
int ar[101],index,count;
for(index=0;index<101;index++)
{
ar[index]=0;
}
for(index=0;index<500;index++)
{
scanf("%d",&count);
ar[count]++;
}
for(index=0;index<101;index++)
{ if(ar[index]>50)
{
printf("Frequency of %d is %d",index,ar[index]);
}
}
}
```

Q12. Write a program to convert a number into its binary using array.

```
#include<stdio.h>
#include<stdlib.h>
int main(){
int a[10],n,i;
printf("Enter the number to convert: ");
scanf("%d",&n);
for(i=0;n>0;i++)
{
a[i]=n%2;
```

```
n=n/2;
}
printf("\nBinary of Given Number is=");
for(i=i-1;i>=0;i–)
{
printf("%d",a[i]);
}
return 0;
}
```

Q13. Write a program to print all negative elements of an array.

```
#include <stdio.h>
#define MAX_SIZE 100 // Maximum array size
void main()
{
int arr[MAX_SIZE]; // Declare array of MAX_SIZE
int index, N;
/* Input size of the array */
printf("Enter size of the array : ");
scanf("%d", &N);
/* Input elements in the array */
printf("Enter elements in array : ");
for(index=0; index<N; index++)
{
scanf("%d", &arr[index]);
}
printf("\nAll negative elements in array are : ");
for(index=0; index<N; index++)
{
/* If current array element is negative */
if(arr[index] < 0)
{
printf("%d\t", arr[index]);
}
}
}
```

Q14. Write a program to count total number of even and odd elements in an array

```
#include <stdio.h>
#define MAX_SIZE 100 //Maximum size of the array
void main()
{
int arr[MAX_SIZE];
int index, size, even, odd;
```

```
/* Input size of the array */
printf("Enter size of the array: ");
scanf("%d", &size);
/* Input array elements */
printf("Enter %d elements in array: ", size);
for(index=0; index<size; index++)
{
scanf("%d", &arr[index]);
}
/* Assuming that there are 0 even and odd elements */
even = 0;
odd = 0;
for(index=0; index<size; index++)
{
/* If the current element of array is even then increment even count */
if(arr[index]%2 == 0)
{
even++;
}
else
{
odd++;
}
}
printf("Total even elements: %d\n", even);
printf("Total odd elements: %d", odd);
}
```

Q15. WAP to count total number of negative elements in array

```
#include <stdio.h>
void main()
{
int arr[100]; // Declares array of size 100
int index, size, count = 0;
/* Input size of array */
printf("Enter size of the array : ");
scanf("%d", &size);
/* Input array elements */
printf("Enter elements in array : ");
for(index=0; index<size; index++)
{
scanf("%d", &arr[index]);
```

```
}
// Count total negative elements in array
for(index=0; index<size; index++)
{
/* Increment count if current array element is negative */
if(arr[index] < 0)
{
count++;
}
}
printf("\nTotal negative elements in array = %d", count);
}
```

Q16. Write a program in C to count a total number of duplicate elements in an array.

```
#include <stdio.h>
void main()
{
   int arr1[100];
        int arr2[100];
        int arr3[100];
   int n,mm=1,ctr=0;
   int i, j;
     printf("\n\nCount total number of duplicate elements in an array:\n");
     printf("------------------------------------------------------\n");
     printf("Input the number of elements to be stored in the array :");
     scanf("%d",&n);
     printf("Input %d elements in the array :\n",n);
     for(i=0;i<n;i++)
      {
            printf("element - %d : ",i);
            scanf("%d",&arr1[i]);
          }
/*----------------- copy in other array ----------------------------------*/
                for(i=0;i<n; i++)
     {
                arr2[i]=arr1[i];
                arr3[i]=0;
     }
/*------------------- mark the elements are duplicate ----------------------*/
        for(i=0;i<n; i++)
     {
                for(j=0;j<n;j++)
                        {
```

```
                        if(arr1[i]==arr2[j])
                        {
                        arr3[j]=mm;
                        mm++;
                        }
                    }
                    mm=1;
    }
/*--------------- Prints the array ------------------------------------*/
  for(i=0; i<n; i++)
   {
    if(arr3[i]==2){ctr++;}
   }
    printf("The total number of duplicate elements found in the array is: %d \n", ctr);
        printf("\n\n");
}
```

Q17. Write a program in C to print all unique elements in an array.

```
#include <stdio.h>
int main()
{
   int arr1[100], n,ctr=0;
   int i, j, k;
     printf("\n\nPrint all unique elements of an array:\n");
     printf("-----------------------------------------\n");
     printf("Input the number of elements to be stored in the array: ");
     scanf("%d",&n);
     printf("Input %d elements in the array :\n",n);
     for(i=0;i<n;i++)
        {
            printf("element - %d : ",i);
            scanf("%d",&arr1[i]);
         }
   printf("\nThe unique elements found in the array are: \n");
   for(i=0; i<n; i++)
   {
      ctr=0;
      for(j=0,k=n; j<k+1; j++)
      {
         /*Increment the counter when the seaarch value is duplicate.*/
         if (i!=j)
         {
                    if(arr1[i]==arr1[j])
          {
```

```
            ctr++;
          }
         }
      }
      if(ctr==0)
      {
        printf("%d ",arr1[i]);
      }
   }
     printf("\n\n");
}
```

Q18. Write a program in C to merge two arrays of same size sorted in descending order.

```
#include <stdio.h>

void main()
{
   int arr1[100], arr2[100], arr3[200];
   int s1, s2, s3;
   int i, j, k;


     printf("\n\nMerge two arrays of same size sorted in decending order.\n");
     printf("-------------------------------------------------------------\n");

     printf("Input the number of elements to be stored in the first array :");
     scanf("%d",&s1);

     printf("Input %d elements in the array :\n",s1);
     for(i=0;i<s1;i++)
        {
           printf("element - %d : ",i);
           scanf("%d",&arr1[i]);
         }


     printf("Input the number of elements to be stored in the second array :");
     scanf("%d",&s2);

     printf("Input %d elements in the array :\n",s2);
     for(i=0;i<s2;i++)
        {
           printf("element - %d : ",i);
           scanf("%d",&arr2[i]);
```

```
            }

   /* size of merged array is size of first array and  size of second array */
   s3 = s1 + s2;
/*----------------- insert in the third array----------------------------------*/
   for(i=0;i<s1; i++)
      {
         arr3[i] = arr1[i];
       }
    for(j=0;j<s2; j++)
      {
         arr3[i] = arr2[j];
         i++;
      }
/*----------------- sort the array in decending order --------------------------*/
  for(i=0;i<s3; i++)
      {
        for(k=0;k<s3-1;k++)
         {

            if(arr3[k]<=arr3[k+1])
             {
               j=arr3[k+1];
               arr3[k+1]=arr3[k];
               arr3[k]=j;
             }
          }
       }

/*---------------- Prints the merged array ----------------------------------*/
    printf("\nThe merged array in decending order is :\n");
   for(i=0; i<s3; i++)
   {
      printf("%d   ", arr3[i]);
   }
         printf("\n\n");
}
```

Q19. Write a program in C to find the second largest element in an array.

```
#include <stdio.h>

void main(){
  int arr1[50],n,i,j=0,lrg,lrg2nd;

      printf("\n\nFind the second largest element in an array :\n");
```

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

    printf("Input the size of array : ");
    scanf("%d", &n);
  /* Stored values into the array*/
    printf("Input %d elements in the array :\n",n);
    for(i=0;i<n;i++)
       {
          printf("element - %d : ",i);
          scanf("%d",&arr1[i]);
        }
/* find location of the largest element in the array */
//   lrg=arr1[0];
  lrg=0;
 for(i=0;i<n;i++)
 {
    if(lrg<arr1[i])
         {
       lrg=arr1[i];
       j = i;
    }
 }

/* ignore the largest element and find the 2nd largest element in the array */
  lrg2nd=0;
 for(i=0;i<n;i++)
 {
   if(i==j)
     {
      i++;  /* ignoring the largest element */
                 i--;
     }
    else
     {
      if(lrg2nd<arr1[i])
          {
        lrg2nd=arr1[i];
       }
     }
 }

 printf("The Second largest element in the array is :  %d \n\n", lrg2nd);
}
```

Q20. Write a program in C to find the second smallest element in an array.

```
#include <stdio.h>

void main()
{
 int arr1[50],n,i,j=0,sml,sml2nd;

     printf("\n\nFind the second smallest element in an array :\n");
     printf("-------------------------------------------------\n");

     printf("Input the size of array : ");
     scanf("%d", &n);
   /* Stored values into the array*/
     printf("Input %d elements in the array (value must be <9999) :\n",n);
     for(i=0;i<n;i++)
        {
           printf("element - %d : ",i);
           scanf("%d",&arr1[i]);
         }
/* find location of the smallest element in the array */
  sml=arr1[0];
 for(i=0;i<n;i++)
 {
    if(sml>arr1[i])
          {
       sml=arr1[i];
       j = i;
    }
 }

/* ignore the smallest element and find the 2nd smallest element in the array */
  sml2nd=99999;
 for(i=0;i<n;i++)
 {
   if(i==j)
     {
      i++;  /* ignoring the smallest element */
                 i--;
     }
    else
     {
      if(sml2nd>arr1[i])
           {
         sml2nd=arr1[i];
        }
```

```
        }
    }

    printf("The Second smallest element in the array is :  %d \n\n", sml2nd);
}
```

Arrays (Two Dimension Questions)

Q1. Write a program in C for a 2D array of size 3x3 and print the matrix.

```
#include <stdio.h>
void main()
{
 int arr1[3][3],i,j;

     printf("\n\nRead a 2D array of size 3x3 and print the matrix :\n");
     printf("---------------------------------------------------\n");
   /* Stored values into the array*/
     printf("Input elements in the matrix :\n");
 for(i=0;i<3;i++)
 {
    for(j=0;j<3;j++)
    {
            printf("element - [%d],[%d] : ",i,j);
            scanf("%d",&arr1[i][j]);
    }
 }

 printf("\nThe matrix is : \n");
 for(i=0;i<3;i++)
 {
    printf("\n");
    for(j=0;j<3;j++)
        printf("%d\t",arr1[i][j]);
 }
 printf("\n\n");
}
```

Q2. Write a program in C for addition of two Matrices of same size.

```
#include <stdio.h>
void main()
{
 int arr1[50][50],brr1[50][50],crr1[50][50],i,j,n;
     printf("\n\nAddition of two Matrices :\n");
     printf("-----------------------------\n");
      printf("Input the size of the square matrix (less than 5): ");
     scanf("%d", &n);
   /* Stored values into the array*/
     printf("Input elements in the first matrix :\n");
     for(i=0;i<n;i++)
      {
        for(j=0;j<n;j++)
        {
```

```
                printf("element - [%d],[%d] : ",i,j);
                scanf("%d",&arr1[i][j]);
        }
      }
  printf("Input elements in the second matrix :\n");
     for(i=0;i<n;i++)
     {
        for(j=0;j<n;j++)
        {
                printf("element - [%d],[%d] : ",i,j);
                scanf("%d",&brr1[i][j]);
        }
     }
   printf("\nThe First matrix is :\n");
  for(i=0;i<n;i++)
   {
     printf("\n");
     for(j=0;j<n;j++)
        printf("%d\t",arr1[i][j]);
   }
  printf("\nThe Second matrix is :\n");
  for(i=0;i<n;i++)
   {
     printf("\n");
     for(j=0;j<n;j++)
     printf("%d\t",brr1[i][j]);
   }
/* calculate the sum of the matrix */
   for(i=0;i<n;i++)
      for(j=0;j<n;j++)
         crr1[i][j]=arr1[i][j]+brr1[i][j];
   printf("\nThe Addition of two matrix is : \n");
   for(i=0;i<n;i++){
      printf("\n");
      for(j=0;j<n;j++)
         printf("%d\t",crr1[i][j]);
   }
   printf("\n\n");
}
```

Q3. Write a program in C for subtraction of two Matrices

```
#include <stdio.h>
void main()
{
  int arr1[50][50],brr1[50][50],crr1[50][50],i,j,n;
```

```
    printf("\n\nSubtraction of two Matrices :\n");
    printf("-----------------------------\n");
     printf("Input the size of the square matrix (less than 5): ");
    scanf("%d", &n);
  /* Stored values into the array*/
    printf("Input elements in the first matrix :\n");
    for(i=0;i<n;i++)
     {
        for(j=0;j<n;j++)
        {
                printf("element - [%d],[%d] : ",i,j);
                scanf("%d",&arr1[i][j]);
        }
     }
    printf("Input elements in the second matrix :\n");
    for(i=0;i<n;i++)
     {
        for(j=0;j<n;j++)
        {
                printf("element - [%d],[%d] : ",i,j);
                scanf("%d",&brr1[i][j]);
        }
     }
 printf("\nThe First matrix is :\n");
 for(i=0;i<n;i++)
  {
   printf("\n");
   for(j=0;j<n;j++)
      printf("%d\t",arr1[i][j]);
  }
 printf("\nThe Second matrix is :\n");
 for(i=0;i<n;i++)
  {
   printf("\n");
   for(j=0;j<n;j++)
   printf("%d\t",brr1[i][j]);
  }
/* calculate the subtraction of the matrix */
 for(i=0;i<n;i++)
    for(j=0;j<n;j++)
       crr1[i][j]=arr1[i][j]-brr1[i][j];
 printf("\nThe Subtraction of two matrix is : \n");
 for(i=0;i<n;i++){
```

```
    printf("\n");
    for(j=0;j<n;j++)
        printf("%d\t",crr1[i][j]);
  }
  printf("\n\n");
}
```

Q4. Write a program in C for multiplication of two square Matrices.

$$\begin{bmatrix} a_{11}, a_{12} \\ a_{21}, a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11}, b_{12} \\ b_{21}, b_{22} \end{bmatrix}$$

$$\begin{bmatrix} a_{11} \times b_{11} + a_{12} \times b_{21} , a_{11} \times b_{12} + a_{12} \times b_{22} \\ a_{21} \times b_{11} + a_{22} \times b_{21} , a_{21} \times b_{12} + a_{22} \times b_{22} \end{bmatrix}$$

$$\begin{array}{|c|c|} \hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} \times \begin{array}{|c|c|} \hline 5 & 6 \\ \hline 7 & 8 \\ \hline \end{array}$$

$$\begin{bmatrix} 1 \times 5 + 2 \times 7 , & 1 \times 6 + 2 \times 8 \\ 3 \times 5 + 4 \times 7 , & 3 \times 6 + 4 \times 8 \end{bmatrix} = \begin{bmatrix} 5 + 14, 6 + 16 \\ 15 + 28, 18 + 32 \end{bmatrix}$$

$$= \begin{array}{|c|c|} \hline 19 & 22 \\ \hline 43 & 50 \\ \hline \end{array}$$

```
#include <stdio.h>
void main()
{
 int arr1[50][50],brr1[50][50],crr1[50][50],i,j,k,r1,c1,r2,c2,sum=0;

    printf("\n\nMultiplication of two Matrices :\n");
    printf("--------------------------------\n");

 printf("\nInput the rows and columns of first matrix : ");
 scanf("%d %d",&r1,&c1);
 printf("\nInput the rows and columns of second matrix : ");
 scanf("%d %d",&r2,&c2);
 if(c1!=r2){
   printf("Mutiplication of Matrix is not possible.");
   printf("\nColumn of first matrix and row of second matrix must be same.");
 }
```

```
  else
     {
      printf("Input elements in the first matrix :\n");
      for(i=0;i<r1;i++)
       {
          for(j=0;j<c1;j++)
          {
                  printf("element - [%d],[%d] : ",i,j);
                  scanf("%d",&arr1[i][j]);
          }
       }
      printf("Input elements in the second matrix :\n");
      for(i=0;i<r2;i++)
       {
          for(j=0;j<c2;j++)
          {
                  printf("element - [%d],[%d] : ",i,j);
                  scanf("%d",&brr1[i][j]);
          }
       }
           printf("\nThe First matrix is :\n");
                  for(i=0;i<r1;i++)
                  {
                  printf("\n");
                  for(j=0;j<c1;j++)
          printf("%d\t",arr1[i][j]);
                  }

          printf("\nThe Second matrix is :\n");
                  for(i=0;i<r2;i++)
                  {
                  printf("\n");
                  for(j=0;j<c2;j++)
                  printf("%d\t",brr1[i][j]);
                  }
//multiplication of matrix
     for(i=0;i<r1;i++)
        for(j=0;j<c2;j++)
         crr1[i][j]=0;
          for(i=0;i<r1;i++)    //row of first matrix
             {
              for(j=0;j<c2;j++)    //column of second matrix
                {
                 sum=0;
```

```
                for(k=0;k<c1;k++)
                  sum=sum+arr1[i][k]*brr1[k][j];
                  crr1[i][j]=sum;
              }
            }
 printf("\nThe multiplication of two matrices is : \n");
 for(i=0;i<r1;i++)
   {
     printf("\n");
     for(j=0;j<c2;j++)
      {
       printf("%d\t",crr1[i][j]);
      }
   }
 }
printf("\n\n");
}
```

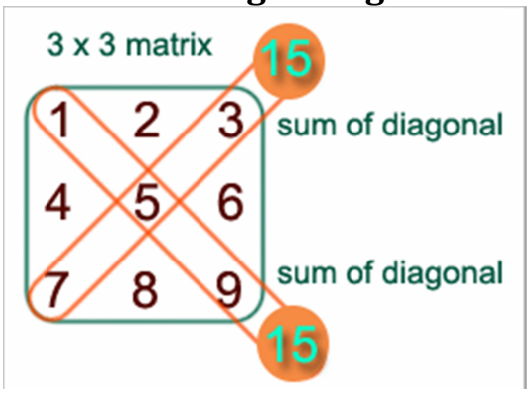
Q5. Write a program in C to find transpose of a given matrix.

```
#include <stdio.h>
void main()
 {
 int arr1[50][50],brr1[50][50],i,j,r,c;
     printf("\n\nTranspose of a Matrix :\n");
     printf("--------------------------\n");
     printf("\nInput the rows and columns of the matrix : ");
     scanf("%d %d",&r,&c);
     printf("Input elements in the first matrix :\n");
     for(i=0;i<r;i++)
      {
         for(j=0;j<c;j++)
         {
                 printf("element - [%d],[%d] : ",i,j);
                 scanf("%d",&arr1[i][j]);
         }
      }
          printf("\nThe matrix is :\n");
                  for(i=0;i<r;i++)
                  {
                  printf("\n");
                  for(j=0;j<c;j++)
         printf("%d\t",arr1[i][j]);
                  }
 for(i=0;i<r;i++)
   {
```

```
    for(j=0;j<c;j++)
        {
            brr1[j][i]=arr1[i][j];
        }
    }

    printf("\n\nThe transpose of a matrix is : ");
    for(i=0;i<c;i++){
    printf("\n");
    for(j=0;j<r;j++){
       printf("%d\t",brr1[i][j]);
    }
 }
    printf("\n\n");
}
```

Q6. Write a program in C to find sum of right diagonals of a matrix.




```
#include <stdio.h>
void main()
  {
   int i,j,arr1[50][50],sum=0,n;
     printf("\n\nFind sum of right diagonals of a matrix :\n");
     printf("---------------------------------------\n");
          printf("Input the size of the square matrix : ");
    scanf("%d", &n);
          printf("Input elements in the first matrix :\n");
     for(i=0;i<n;i++)
      {
        for(j=0;j<n;j++)
        {
                printf("element - [%d],[%d] : ",i,j);
                scanf("%d",&arr1[i][j]);
                        if (i==j) sum= sum+arr1[i][j];
        }
      }
```

```
        printf("The matrix is :\n");
        for(i=0;i<n;i++)
        {
         for(j=0;j<n ;j++)
           printf("% 4d",arr1[i][j]);
          printf("\n");
        }
    printf("Addition of the right Diagonal elements is :%d\n",sum);
  }
```

Q7. Write a program in C to find sum of rows an columns of a Matrix

```
#include <stdio.h>
void main()
{
   int i,j,k,arr1[10][10],rsum[10],csum[10],n;
     printf("\n\nFind the sum of rows an columns of a Matrix:\n");
     printf("-----------------------------------------\n");
   printf("Input the size of the square matrix : ");
   scanf("%d", &n);
        printf("Input elements in the first matrix :\n");
    for(i=0;i<n;i++)
     {
        for(j=0;j<n;j++)
        {
                printf("element - [%d],[%d] : ",i,j);
                scanf("%d",&arr1[i][j]);
        }
     }
        printf("The matrix is :\n");
        for(i=0;i<n;i++)
        {
         for(j=0;j<n ;j++)
           printf("% 4d",arr1[i][j]);
          printf("\n");
        }
   /* Sum of rows */
    for(i=0;i<n;i++)
    {
         rsum[i]=0;
         for(j=0;j<n;j++)
         rsum[i]=rsum[i]+arr1[i][j];
    }
     /* Sum of Column */
     for(i=0;i<n;i++)
```

```
    {
          csum[i]=0;
          for(j=0;j<n;j++)
                csum[i]=csum[i]+arr1[j][i];
    }
    printf("The sum or rows and columns of the matrix is :\n");
    for(i=0;i<n;i++)
    {
          for(j=0;j<n;j++)
            printf("% 4d",arr1[i][j]);
          printf("% 8d",rsum[i]);
          printf("\n");
     }
     printf("\n");
           for(j=0;j<n;j++)
         {
              printf("% 4d",csum[j]);
         }
        printf("\n\n");
 }
```

Q8. Write a program in C to print or display the lower triangular of a given matrix.

```
#include <stdio.h>
void main()
  {
 int arr1[10][10],i,j,n;
 float determinant=0;
     printf("\n\nDisplay the lower triangular of a given matrix :\n");
     printf("-----------------------------------------------------\n");
    printf("Input the size of the square matrix : ");
    scanf("%d", &n);
          printf("Input elements in the first matrix :\n");
     for(i=0;i<n;i++)
      {
         for(j=0;j<n;j++)
         {
                printf("element - [%d],[%d] : ",i,j);
                scanf("%d",&arr1[i][j]);
         }
      }
          printf("The matrix is :\n");
          for(i=0;i<n;i++)
          {
           for(j=0;j<n ;j++)
             printf("% 4d",arr1[i][j]);
```

```
            printf("\n");
          }
  printf("\nSetting zero in lower triangular matrix\n");
  for(i=0;i<n;i++){
    printf("\n");
    for(j=0;j<n;j++)
        if(i<=j)
          printf("% 4d",arr1[i][j]);
        else
          printf("% 4d",0);
 }
     printf("\n\n");
}
```

Q9. Write a program in C to print or display upper triangular matrix.

```
#include <stdio.h>
void main()
 {
  int arr1[10][10],i,j,n;
  float determinant=0;
 printf("\n\nDisplay the upper triangular of a given matrix :\n");
      printf("-------------------------------------------\n");
   printf("Input the size of the square matrix : ");
    scanf("%d", &n);
          printf("Input elements in the first matrix :\n");
     for(i=0;i<n;i++)
      {
         for(j=0;j<n;j++)
         {
                 printf("element - [%d],[%d] : ",i,j);
                 scanf("%d",&arr1[i][j]);
         }
      }
          printf("The matrix is :\n");
          for(i=0;i<n;i++)
          {
            for(j=0;j<n ;j++)
              printf("% 4d",arr1[i][j]);
            printf("\n");
          }
  printf("\nSetting zero in upper triangular matrix\n");
  for(i=0;i<n;i++)
   {
    printf("\n");
    for(j=0;j<n;j++)
```

```
        if(i>=j)
          printf("% 4d",arr1[i][j]);
        else
          printf("% 4d",0);
 }
    printf("\n\n");
}
```

Q10. Write a program in C to calculate determinant of a 3 x 3 matrix.

```
#include <stdio.h>
void main()
  {
 int arr1[10][10],i,j,n;
 int det=0;
 printf("\n\nCalculate the determinant of a 3 x 3 matrix :\n");
    printf("-----------------------------------------------\n");
printf("Input elements in the first matrix :\n");
    for(i=0;i<3;i++)
     {
        for(j=0;j<3;j++)
        {
                printf("element - [%d],[%d] : ",i,j);
                scanf("%d",&arr1[i][j]);
        }
     }
         printf("The matrix is :\n");
         for(i=0;i<3;i++)
         {
          for(j=0;j<3 ;j++)
            printf("% 4d",arr1[i][j]);
           printf("\n");
         }
 for(i=0;i<3;i++)
    det = det + (arr1[0][i]*(arr1[1][(i+1)%3]*arr1[2][(i+2)%3] -
        arr1[1][(i+2)%3]*arr1[2][(i+1)%3]));
 printf("\nThe Determinant of the matrix is: %d\n\n",det);
}
```

Q11. Write a program in C to accept a matrix and determine whether it is a sparse matrix.

```
#include <stdio.h>
/*A sparse martix is matrix which  has more zero elements than nonzero elements */
void main ()
{
        static int arr1[10][10];
        int i,j,r,c;
```

```
        int ctr=0;
    printf("\n\nDetermine whether a matrix is a sparse matrix :\n");
    printf("-------------------------------------------------\n");
    printf("Input the number of rows of the matrix : ");
    scanf("%d", &r);
    printf("Input the number of columns of the matrix : ");
    scanf("%d", &c);
         printf("Input elements in the first matrix :\n");
      for(i=0;i<r;i++)
       {
          for(j=0;j<c;j++)
          {
                  printf("element - [%d],[%d] : ",i,j);
                  scanf("%d",&arr1[i][j]);
                          if (arr1[i][j]==0)
                          {
                                  ++ctr;
                          }
          }
       }
          if (ctr>((r*c)/2))
          {
                   printf ("The given matrix is sparse matrix. \n");
          }
          else
                   printf ("The given matrix is not a sparse matrix.\n");

          printf ("There are %d number of zeros in the matrix.\n\n",ctr);
}
```

Q12. Write a program in C to accept two matrices and check whether they are equal.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
  int arr1[50][50], brr1[50][50];
  int i, j, r1, c1, r2, c2, flag =1;
     printf("\n\nAccept two matrices and check whether they are equal :\n ");
     printf("-------------------------------------------------------\n");
  printf("Input Rows and Columns of the 1st matrix :");
  scanf("%d %d", &r1, &c1);
  printf("Input Rows and Columns of the 2nd matrix :");
  scanf("%d %d", &r2,&c2);
         printf("Input elements in the first matrix :\n");
     for(i=0;i<r1;i++)
```

```
    {
       for(j=0;j<c1;j++)
       {
              printf("element - [%d],[%d] : ",i,j);
              scanf("%d",&arr1[i][j]);
       }
    }
   printf("Input elements in the second matrix :\n");
   for(i=0;i<r2;i++)
    {
       for(j=0;j<c2;j++)
       {
              printf("element - [%d],[%d] : ",i,j);
              scanf("%d",&brr1[i][j]);
       }
    }
        printf("The first matrix is :\n");
        for(i=0;i<r1;i++)
        {
          for(j=0;j<c1 ;j++)
            printf("% 4d",arr1[i][j]);
           printf("\n");
        }
        printf("The second matrix is :\n");
        for(i=0;i<r2;i++)
        {
          for(j=0;j<c2 ;j++)
            printf("% 4d",brr1[i][j]);
           printf("\n");
        }
/* Comparing two matrices for equality */
if(r1 == r2 && c1 == c2)
{
       printf("The Matrices can be compared : \n");
       for(i=0; i<r1; i++)
       {
              for(j=0; j<c2; j++)
              {
                     if(arr1[i][j] != brr1[i][j])
                     {
                            flag = 0;
                            break;
                     }
              }
```

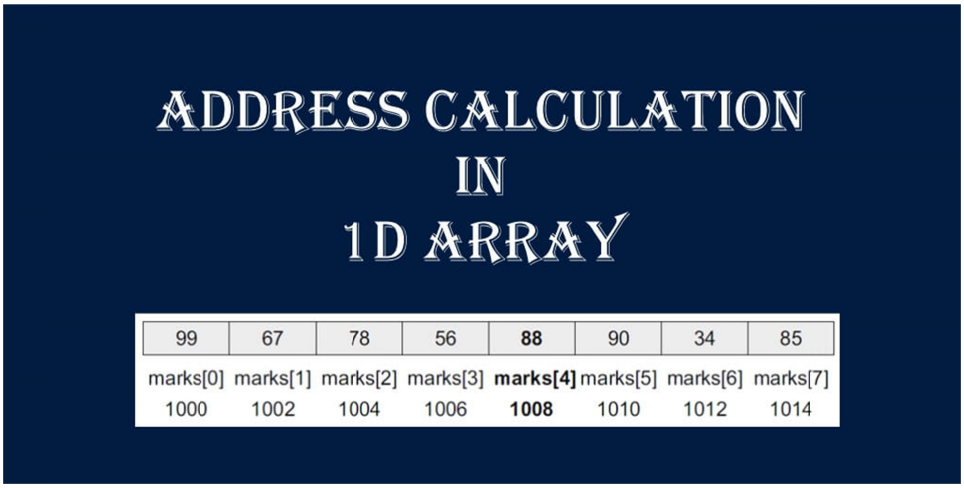
```
        }
    }
    else
    {  printf("The Matrices Cannot be compared :\n");
       exit(1);
    }
    if(flag == 1 )
          printf("Two matrices are equal.\n\n");
    else
          printf("But,two matrices are not equal\n\n");
}
```

Q13. Write a program in C to check whether a given matrix is an identity matrix.

```
#include <stdio.h>
//In a square matrix if all the main diagonal elements are 1's and
//all the remaining elements are 0's is called an Identity Matrix.
void main()
{
  int arr1[10][10];
  int r1,c1;
  int i, j, yn =1;
     printf("\n\n Check whether a given matrix is an identity matrix :\n ");
     printf("------------------------------------------------------------\n");
  printf("Input number of Rows for the matrix :");
  scanf("%d", &r1);
  printf("Input number of Columns for the matrix :");
  scanf("%d",&c1);
         printf("Input elements in the first matrix :\n");
     for(i=0;i<r1;i++)
      {
         for(j=0;j<c1;j++)
         {
                 printf("element - [%d],[%d] : ",i,j);
                 scanf("%d",&arr1[i][j]);
         }
      }
         printf("The matrix is :\n");
         for(i=0;i<r1;i++)
         {
           for(j=0;j<c1 ;j++)
             printf("% 4d",arr1[i][j]);
           printf("\n");
         }
   for(i=0; i<r1; i++)
   {
```

```
    for(j=0; j<c1; j++)
    {
          if(arr1[i][j] != 1 && arr1[j][i] !=0)
          {
            yn = 0;
            break;
          }
         }
   }
   if(yn == 1 )
         printf(" The matrix is an identity matrix.\n\n");
   else
         printf(" The matrix is not an identity matrix.\n\n");
}
```

Address Calculation in 1d Array with example



Array

An array is a collection of similar data elements. These data elements have the same data type. The elements of the array are stored in consecutive memory locations and are referenced by an index (also known as the subscript). The subscript is an ordinal number which is used to identify an element of the array.

Ex: int A[5];

Calculating the Address of 1D Array Elements

- The array name is a symbolic reference to the address of the first byte of the array. When we use the array name, we are actually referring to the first byte of the array.
- The subscript or the index represents the offset from the beginning of the array to the element being referenced. That is, with just the array name and the index, C can calculate the address of any element in the array.
- Since an array stores all its data elements in consecutive memory locations, storing just the base address, that is the address of the first element in the array, is sufficient. The address of other data elements can simply be calculated using the base address.
- The formula to perform this calculation is,

Address of A[k] = Base_Address(A) + w(k – Lower_Bound)

Here, A is the array, k is the index of the element of which we have to calculate the address, Base_Address is the base address of the array A, and w is the size of one element in memory, for example, size of int is 2 bytes.

Address calculation in 1d array : Example

***Example 1** : Given an array int marks[] = {99,67,78,56,88,90,34,85}, calculate the address of marks[4] if the base address = 1000.*

Solution :

99	67	78	56	**88**	90	34	85
marks[0]	marks[1]	marks[2]	marks[3]	**marks[4]**	marks[5]	marks[6]	marks[7]
1000	1002	1004	1006	**1008**	1010	1012	1014

We know that storing an integer value requires 2 bytes, therefore, its size is 2 bytes.

Address of A[k] = Base_Address(A) + w(k – Lower_Bound)

marks[4] = 1000 + 2(4 – 0)
= 1000 + 2(4)
= 1008 **[Ans]**

Example 2** : Base address of an array **B[1300 : 1900]** as 1020 and size of each element of array is 2 bytes in the memory. Find the address of **B[1700].

Solution :

The given values are: Base_Address = 1020, Lower_Bound = 1300, w = 2, k = 1700

Address of A[k] = Base_Address(A) + w(k – Lower_Bound)

$$\begin{aligned} \textbf{B[1700]} &= 1020 + 2 * (1700 - 1300) \\ &= 1020 + 2 * 400 \\ &= 1020 + 800 \\ &= 1820 \textbf{ [Ans]} \end{aligned}$$

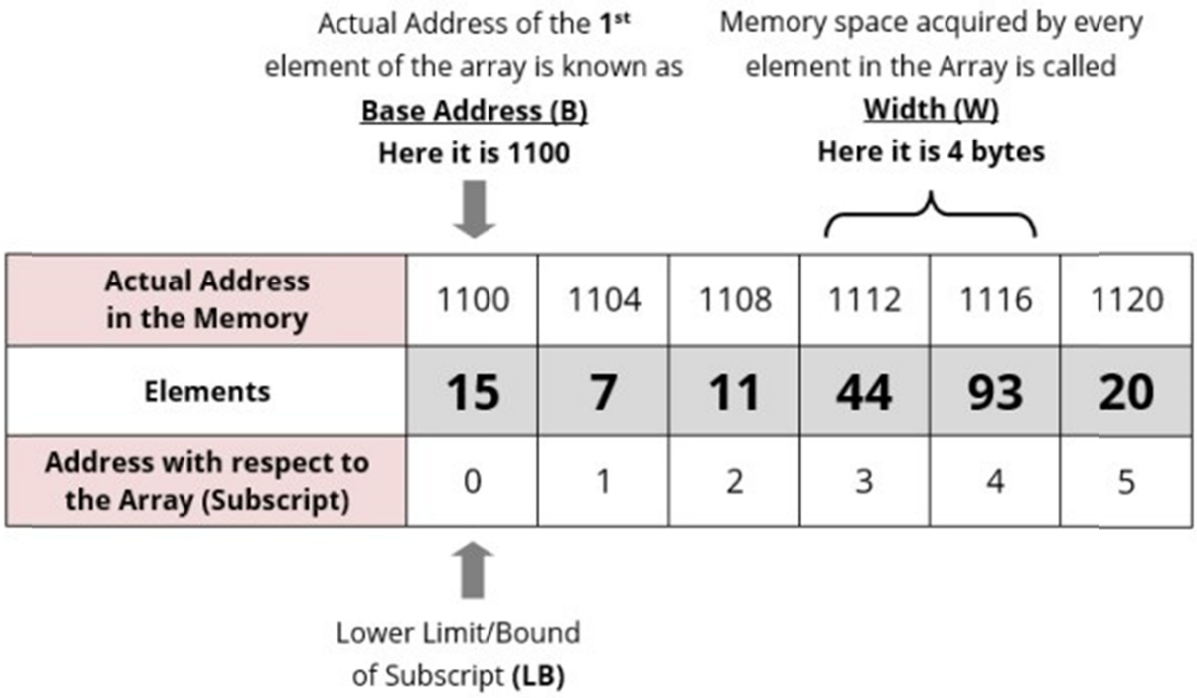
Calculating the Length of an Array

The length of an array is given by the number of elements stored in it. The general formula to calculate the length of an array is

$$\textbf{Length = upper\_bound – lower\_bound + 1}$$

where upper_bound is the index of the last element and lower_bound is the index of the first element in the array.

Address Calculation in single (one) Dimension Array:



Array of an element of an array say “A[I]” is calculated using the following formula:

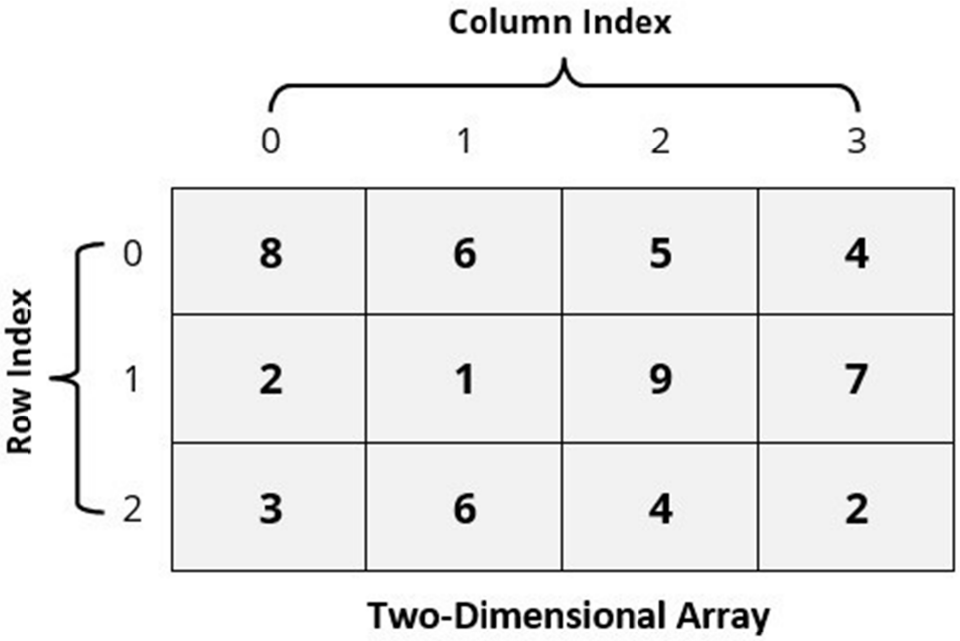
Address of A [I] = B + W * (I – LB)
Where,
B = Base address
W = Storage Size of one element stored in the array (in byte)
I = Subscript of element whose address is to be found
LB = Lower limit / Lower Bound of subscript, if not specified assume 0 (zero)
Example:
Given the base address of an array **B[1300…..1900]** as 1020 and size of each element is 2 bytes in the memory. Find the address of **B[1700]**.
Solution:
The given values are: B = 1020, LB = 1300, W = 2, I = 1700

Address of A [I] = B + W * (I – LB)

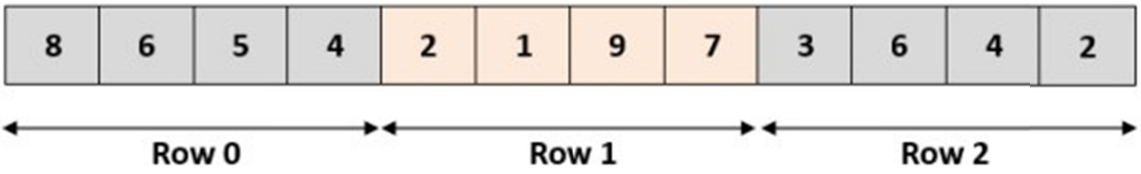
= 1020 + 2 * (1700 – 1300)
= 1020 + 2 * 400
= 1020 + 800
= 1820 **[Ans]**

Address Calculation in Double (Two) Dimensional Array:

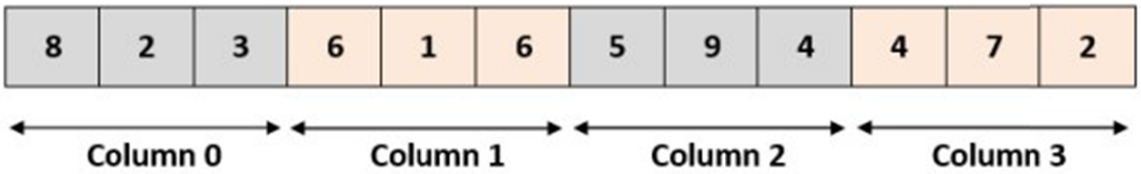
While storing the elements of a 2-D array in memory, these are allocated contiguous memory locations. Therefore, a 2-D array must be linearized so as to enable their storage. There are two alternatives to achieve linearization: Row-Major and Column-Major.



Row-Major (Row Wise Arrangement)



Column-Major (Column Wise Arrangement)



Address of an element of any array say "**A[I][J]**" is calculated in two forms as given: (1) Row Major System (2) Column Major System

Row Major System:

The address of a location in Row Major System is calculated using the following formula:

Address of A [I][J] = B + W * [N * (I – Lr) + (J – Lc)]

Column Major System:

The address of a location in Column Major System is calculated using the following formula:

Address of A [I][J] Column Major Wise = B + W * [(I – Lr) + M * (J – Lc)]

Where,
B = Base address
I = Row subscript of element whose address is to be found
J = Column subscript of element whose address is to be found
W = Storage Size of one element stored in the array (in byte)
Lr = Lower limit of row/start row index of matrix, if not given assume 0 (zero)
Lc = Lower limit of column/start column index of matrix, if not given assume 0 (zero)
M = Number of row of the given matrix
N = Number of column of the given matrix
Important : Usually number of rows and columns of a matrix are given (like A[20][30] or A[40][60]) but if it is given as **A[Lr- – – – – Ur, Lc- – – – – Uc]**. In this case number of rows and columns are calculated using the following methods:
Number of rows (**M**) will be calculated as = **(Ur – Lr) + 1**
Number of columns (**N**) will be calculated as = **(Uc – Lc) + 1**
And rest of the process will remain same as per requirement (Row Major Wise or Column Major Wise).

Examples:
Q 1. An array X [-15…….10, 15……40] requires one byte of storage. If beginning location is 1500 determine the location of X [15][20].
Solution:
As you see here the number of rows and columns are not given in the question. So they are calculated as:

Number or rows say **M = (Ur – Lr) + 1** = [10 – (- 15)] +1 = 26
Number or columns say **N = (Uc – Lc) + 1** = [40 – 15)] +1 = 26
(i) Column Major Wise Calculation of above equation
The given values are: B = 1500, W = 1 byte, I = 15, J = 20, Lr = -15, Lc = 15, M = 26

Address of A [I][J] = B + W * [(I – Lr) + M * (J – Lc)]

= 1500 + 1 * [(15 – (-15)) + 26 * (20 – 15)] = 1500 + 1 * [30 + 26 * 5] = 1500 + 1 * [160] = 1660 **[Ans]**
(ii) Row Major Wise Calculation of above equation
The given values are: B = 1500, W = 1 byte, I = 15, J = 20, Lr = -15, Lc = 15, N = 26

Address of A [I][J] = B + W * [N * (I – Lr) + (J – Lc)]

= 1500 + 1* [26 * (15 – (-15))) + (20 – 15)] = 1500 + 1 * [26 * 30 + 5] = 1500 + 1 * [780 + 5] = 1500 + 785
= 2285 **[Ans]**