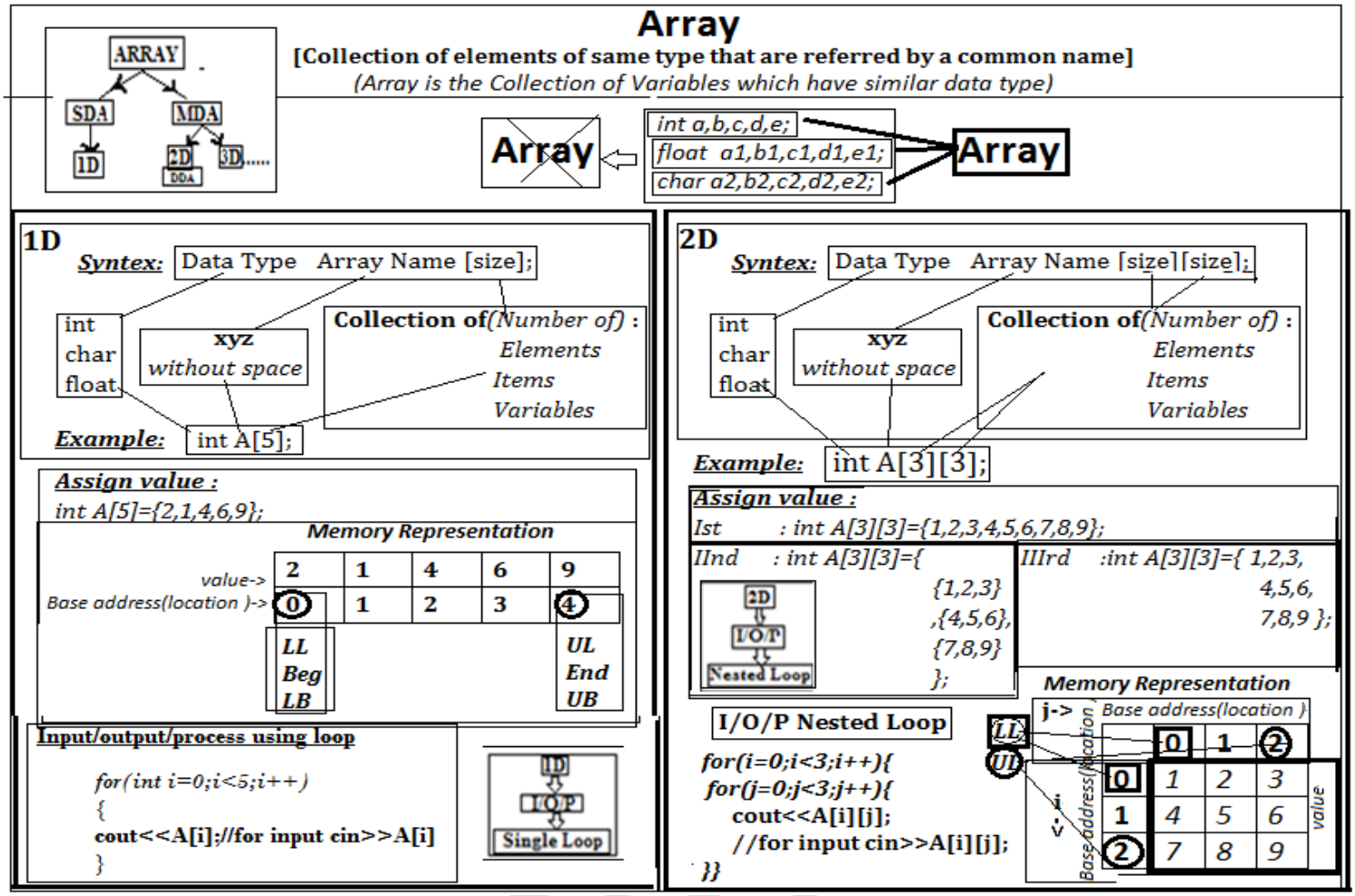


Set 2 1D (SDA) Array Assignment

Q1. WAP to **input** and **display** element of SDA **size 16.**
Q2. WAP to **add** all elements of SDA **size 16.**
Q3. WAP to **multiply** all element of SDA **size 12.**
Q4. WAP to **displays square** of all elements in SDA **size 16.**
Q5. WAP to **displays** all elements even position of SDA **size 16.**
Q6. WAP to **displays** all elements odd position of SDA **size 16.**
Q7. WAP **square** all element of SDA **size 16**
Q8. WAP to **find even and odd** elements in SDA **size 16.**
Q9. WAP to **count even and odd** elements in SDA **size 16.**
Q10. WAP to **add all even and odd** elements of SDA **size 16.**
Q11. WAP to **replace even by 0 odd by 1** element in SDA **size 16.**
Q12. WAP to **find** an element in SDA **size 16.**
Q13. WAP to **count** an element in SDA **size 16.**
Q14. WAP to **replace** an element in SDA by 0 **size 16.**
Q15. WAP to change the search elements **size 16.**
Q16. WAP to **find** prime no. in SDA elements **size 16.**
Q17. WAP to **count** prime no. in SDA elements **size 16.**
Q18. WAP to **replace by 0** prime no. in SDA elements **size 16.**
Q19. WAP to add all prime no. of SDA **size 16.**
Q20. WAP to **reverse an** SDA **size 16.**
Q21. WAP to **reverse then display** SDA **size 16.**
Q22. WAP to multiply even by 2 and odd by 3 in SDA **size 16.**

Set 3 2D (DDA) Array Assignment

Q1. WAP **Accept** DDA and display **size 3*3.**
Q2. WAP to **add** all element of DDA **size 3*3.**
Q3. WAP to **Multiply** all element of DDA **size 3*3.**
Q4. WAP to **displays square** of all elements in DDA **size 3*3.**
Q5. WAP to **displays** all elements even position of DDA **size 3*3.**
Q6. WAP to **displays** all elements odd position of DDA **size 3*3.**
Q7. WAP **square** all element of DDA **size 3*3.**
Q8. WAP to **find even and odd** elements in DDA **size 3*3.**
Q9. WAP to **count even and odd** elements in DDA **size 3*3.**
Q10. WAP to **add all even and odd** elements of DDA **size 3*3.**
Q11. WAP to **replace even by 0 odd by 1** element in DDA **size 3*3.**
Q12. WAP to **find** an element in DDA **size 3*3.**
Q13. WAP to **count** an element in DDA **size 3*3.**
Q14. WAP to **replace** an element in DDA **size 3*3.**
Q15. WAP to **find** prime no. in DDA elements **size 3*3.**
Q16. WAP to **count** prime no. in DDA elements **size 3*3.**
Q17. WAP to **replace by 0** prime no. in DDA elements **size 3*3.**
Q18. WAP to add all prime no. of DDA **size 3*3.**
Q19. WAP to **display square of** all elements **size 3*3.**
Q20. WAP to **change the search** elements **size 3*3.**
Q21. WAP to display all right diagonal no. of DDA size 3*3
Q22. WAP to display all left diagonal no. of DDA size 3*3
Q23. WAP to add all right diagonal no. of DDA size 3*3
Q24. WAP to add all left diagonal no. of DDA size 3*3
Q25. WAP to replace all right diagonal no. by 0 of DDA size 3*3
Q26. WAP to replace all left diagonal no. by 1 of DDA size 3*3
Q27. WAP to replace all right diagonal no. by 0 of DDA size 3*3
Q28. WAP to replace all left diagonal no. by 1 of DDA size 3*3

One Dimensional array

An array is a continuous memory location holding similar type of data in single row orsingle column. Declaration in c++ is as under:

Declaration of Array: Type arrayName [NumberOfElements];

For example:

int Age[5] ;

int A[5] = {11,2,23,4,15};

It is possible to leave the array size open. The compiler will count the array size.int B[] = {6,7,8,9,15,12};

Using Loop to input an Array from user

```
int age[50],I;
for (i=0; i<50; i++)P
{
 cin>>age[i];
}
```

• **String (Array of characters)** –Defined in c++ as one dimensional array of characters as:

char s[80]= "Object oriented programming";

Multi Dimensional Arrays

Two Dimensional Array: It is a collection of data elements of same data type arranged in rows and columns (that is, in two dimensions).

Declaration of Two-Dimensional Array :

Type ArrayName[NumberOfRows][NumberOfColumn];

For example: `int Sales[3][5];`

Initialization of Two-Dimensional Array

All rows are enclosed within curly braces.int A[4][3] = {{22, 23, 10}, {15, 25, 13},{20, 74, 67},{11, 18, 14}};

Referring to Array Elements

The format is as simple as:name[rowIndex][columnIndex]

Examples:

```
cout<<A[1][2];     //print an array element
A[1][2]=13;       // assign value to an array element
cin>>A[1][2];      //input element
```

Using Loop to input an Two-Dimensional Array from user

```
int mat[3][5], row, col ;
for (row = 0; row < 3; row++)
      for (col = 0; col < 5; col++)
      cin >> mat[row][col];
```

ID Arrays as Parameters

At some moment we may need to pass an array to a function as a parameter. In C++ it is not possible to pass a complete block of memory by value as a parameter to a function, but we are allowed to pass its address.

For example, the following function:

```
void print(int A[])
```

accepts a parameter of type "array of int" called A.

In order to pass to this function an array declared as:

```
int arr[20];
```

we need to write a call like this: print(arr);

Here is a complete example:

```
#include <iostream>
//using namespace std;
void print(int A[], int length,)
{
 for (int n=0; n<length; n++)
   cout << A[n] << " ";
 cout << "\n";
}
int main ()
{
 int arr[] = {5, 10, 15};
 print(arr,3);
 return 0;
}
```

Basic Operation On One Dimensional ArrayFunction to traverse the array A

```
void display(int A[], int n)
{
    cout<<"The elements of the array are:\n";
    for(int i=0;i<n;i++)
        cout<<A[i];
}
```

Function to Read elements of the array A

```
void Input(int A[], int n)
{
    cout<<"Enter the elements:";
    for(int i=0;i<n;i++)
        cin>>A[i];
}
```

2D Arrays as Parameters

Two-dimensional arrays can be passed as parameters to a function, and they are passed by reference. When declaring a two-dimensional array as a formal parameter, we can omit the size of the first dimension,

but not the second; that is, we must specify the number of columns. For example:

```
void print(int A[][3],int N, int M)
```

In order to pass to this function an array declared as:

```
int arr[4][3];
```

we need to write a call like this: `print(arr);`

Here is a complete example:

```
#include <iostream>
using namespace std;
void print(int A[][3],int N, int M)
{
 for (R = 0; R < N; R++)p
  for (C = 0; C < M; C++)
    cout << A[R][C];
}
int main ()
{
 int arr[4][3] ={{12, 29, 11},
                 {25, 25, 13},
                 {24, 64, 67},
                 {11, 18, 14}};
 print(arr,4,3);
 return 0;
}
```

Basic Operation On two Dimensional ArrayFunction

Function to read the array A

```
void Read(int A[][20], int N, int M)
{
 for(int R=0;R<N;R++)
  for(int C=0;C<M;C++)
  {
   cout<<"(R<<','<<")?";
   cin>>A[R][C];
  }
}
```

Function to display content of a two dimensional array A

```
void Display(int A[][20],int N, int M)
{
 for(int R=0;R<N;R++)
 {
  for(int C=0;C<M;C++)
    cout<<setw(10)<<A[R][C];
  cout<<endl;
 }
}
```

Function to find the sum of two dimensional arrays A and B

```
void Addition(int A[][20], int B[][20],int N, int M)
{
 for(int R=0;R<N;R++)
  for(int C=0;C<M;C++)
   C[R][C]=A[R][C]+B[R][C];
}
```

Function to multiply two dimensional arrays A and B of order NxL and LxM

```
void Multiply(int A[][20], int B[][20], int C[][20],int N,
int L, int M)
{
 for(int R=0;R<N;R++)
  for(int C=0;C<M;C++)
  {
   C[R][C]=0;
   for(int T=0;T<L;T++)
    C[R][C]+=A[R][T]*B[T][C];
  }
}
```

Function to find & display sum of rows & sum of cols. of a 2 dim. array A

```
void SumRowCol(int A[][20], int N, int M)
{
 for(int R=0;R<N;R++)
 {
  int SumR=0;
  for(int C=0;C<M;C++)
   SumR+=A[R][C];
  cout<<"Row("<<R<<")="<<SumR<<endl;
 }
 for(int R=0;R<N;R++)
 {
  int SumR=0;
  for(int C=0;C<M;C++)
   SumR+=A[R][C];
  cout<<"Row("<<R<<")="<<SumR<<endl;
 }
}
```

Function to find sum of diagonal elements of a square matrix A

```
void Diagonal(int A[][20], int N, int &Rdiag, int
&LDiag)
{
 for(int I=0,Rdiag=0;I<N;I++)
  Rdiag+=A[I][I];
 for(int I=0,Ldiag=0;I<N;I++)
  Ldiag+=A[N-I-1][I];
}
```

Function to find out transpose of a two dimensional array A

```
void Transpose(int A[][20], int B[][20],int N, int M)
{
 for(int R=0;R<N;R++)
  for(int C=0;C<M;C++)
    B[R][C]=A[C][R];
}
```

Set1

1.int i,a[5]={5,4,3,2,1};
for(i=0;i<5;i++)
{
if(a[i]%2= =0)a[i]+=2;
else a[i]+=3;
cout<<a[i];
}

2.int i,a[5]={1,2,3,4,5};
for(i=0;i<=4;i++)
{
a[i]=a[i]*2+i*2;
cout<<a[i];
}

1. 8 6 6 4 4
2. 2 6 10 14 18
3. 3 6

3.int i,a[7]={1,2,3,4,5,6,7};
for(i=0;i<=6;i++)
{
if(a[i]%3==0)
cout<<a[i];}

Find output

4.int i,a[9]={1,2,3,4,5,6,7,8,9};
for(i=0;i<=6;i++)
{
if(a[i]%4==0)
cout<<a[i];
}

5.int i,a[9]={1,2,3,4,5,6,7,8,9};
for(i=0;i<=6;i++)
{
if(a[i]%2==0)
cout\<<a[i]+3;
else
cout<<a[i];
}

4. 4
5. 4 1 5 3 7 5 9 7

6.int i,a[5]={1,2,3,4,5};
for(i=2;i<=4;i++)
{cout<<a[i]*5;}

7.int i,a[8];

6. 15 20 25
7. 11 16 13 47 7

a[8]={11,21,13,42,35,12,34,65};
for(i=0;i<=4;i++)
{
if(a[i]%2==0) a[i]+=5;
if(a[i]%3==0)a[i]-=5;
if(a[i]%5==0)a[i]/=5;
cout<<a[i];
}

8.int i,a[8] ,t=0,n=8;;
a[8]={11,21,13,42,35,12,34,65};
int l=n-1;
for(i=0;i<n/2;i++)
{
t=a[l]-1;
a[l]=a[i]*2;
a[i]=t;}
for(i=0;i<n;i++)}

8. 64 21 41 25 35 12 34 84

Searching Sorting

Linear Search:

```
int linear_search(int a[],int n, int item)
{
for(int i=0; i<n; i++)
{ if(a[i]==item)
return i;
}
```

Binary Search

```
int binary_search(int a[],int n, int
item)
{int beg,mid,last;
beg=0;
last=n-1;
while(beg<=last)
{
 mid = (beg + last)/2;
if(item==a[i])
return mid;
else if(item>a[mid])
beg=mid+1;
else
last=mid-1;
}
return -1;
}
```

Sorting

Sorting means arranging the elements in some specific order, i.e. either ascending or descending order. The various sorting techniques available are

(i) **Insertion Sort**: Initially, the first element is assumed to be sorted. In the first pass, 2nd element is inserted into its proper place in the sorted part of the array. Similarly in the next pass, the 3rd element is placed and so on. Given below is the insertion sort for ascending order.

Array at beginning:	42	29	74	11	65	58
After pass 1	29	42	74	11	65	58
After pass2	29	42	74	11	65	58
After pass 3	11	29	42	74	65	58
After pass 4	11	29	42	65	74	58
After pass 5	11	29	42	58	65	74
Sorted Array	**11**	**29**	**42**	**58**	**65**	**74**

```
//function for Insertion Sort
void InsertionSort(int a[],int n)
{ int i,j,temp;
for(i=1;i<n;i++)
{ temp=a[i];
j=i-1;
```

```
while(temp<a[j])&&j>=0)
{a[j+1]=a[j];
j--;
}
a[j+1]=temp;
cout<,"\n After Pass "<<i ;
for(k=0;k<n;k++)
cout<<a[k]; } }
```

(ii) **Selection Sort**: The element with the smallest value (if found) is swapped with the first element. As a result of this interchange, the smallest element is placed in the 1st position of the array. In the second pass, second smallest element is searched and swapped with second element and so on. Given below is the selection sort for ascending order.

Array at beginning:	42	29	74	11	65	58
After pass 1	**11**	29	74	**42**	65	58
After pass2	11	**29**	74	42	65	58
After pass 3	11	29	**42**	**74**	65	58
After pass 4	11	29	42	**58**	65	**74**
After pass 5	11	29	42	58	**65**	74
Sorted Array	**11**	**29**	**42**	**58**	**65**	**74**

```
//function for Selection Sort
void SelectionSort(int a[],int n)
{ int i,small,pos,temp;
for(i=0;i<n;i++)
{ small=a[i];
pos=i;
for(j=i+1;j<n;j++)
{if(a[j]<small)
{small=a[j];
Pos=j;}
}
temp=a[i];
a[i]=a[pos];
a[pos]=temp;
cout<,"\n After Pass "<<i+1 ;
for(j=0;j<n;j++)
cout<<a[j];
}
}
```

(iii) **Bubble Sort**: In this technique, two adjacent values are compared and they are exchanged if not in proper order. In every pass, the larger element settles at its appropriate position in the bottom. Given below is the bubble sort for ascending order.

Array at beginning:	42	29	74	11	65	58
After pass 1	29	42	11	65	58	**74**
After pass2	29	11	42	58	**65**	74
After pass 3	11	29	42	**58**	65	74
After pass 4	11	29	**42**	58	65	74
After pass 5	11	**29**	42	58	65	74
Sorted Array	**11**	**29**	**42**	**58**	**65**	**74**

```
//function for Bubble Sort
void BubbleSort(int a[],int n)
{ int temp;
for(int i=0;i<n;i++)
{ for(int j=0;j<n-i;j++)
{if(a[j]>a[j+1])
{temp=a[j];
A[j]=a[j+1];
A[j+1]=temp;
} }
cout<,"\n After Pass "<<i+1 ;
for(int k=0;k<n;k++)
cout<<a[k];
} }
```

(iv) **Merge Sort**: Merging is the process of combining two or more sorted arrays into another array which is also sorted. In merge sort the array is sorted while merging.

Suppose we have to merge array A and array B. The first element of array A is compared with the first element of array B. If the first element of array A is smaller than the first element of array B, the element from array A is moved to the new array C. The subscript of array A is now increased since the first element is now set and we move on. If the element from array B should be smaller, it is moved to the new array C. The subscript of array B is increased. This process of comparing the elements in the two arrays continues until either array A or array B is empty. When one array is empty, any elements remaining in the other (non-empty) array are "pushed" into the end of array C and the merge is complete.

```
//function for Merge Sort
void MergeSort(int a[],int b[],int c[],int
m,int n)
{ int i=0,j=0,k=0,r;
while(i<m &&j<n)
{ if(a[i]<=b[j])
{c[k]=a[i];
i++;
}
else
{c[k]=b[j];
j++;
}
k++;
}
if(i==m)
{for(r=j;r<n;r++)
{ c[k]=b[r];
k++;} }
else
{
for(r=i;r<m;r++)
{
 c[k]=a[r];
k++; } }
```

Set 4

1. From a 2D array ARR[3][3] WAP to prepare one dimensions array ARR2[9] that will have all the elements of ARR as if they are stored in row major form.
   a. 1 2 3
   b. 4 5 6
   c. 7 8 9 then should contain 123456789
2. Write program in C++ to print the row sum and column sum of a matrix
3. Write program to display sum of **upper half** elements of a square matrix. The function will return the sum of the values.

1	2	3
4	5	6
7	8	9

R1=1+2+3+4+5+7, R2=1+4+5+7+8+9, R3=1+2+3+5+6+9, R4=3+5+6+7+8+9

4. WAP in C++ which accepts a 2D array of integers and its size as arguments and displays the elements of middle row and the elements of middle column. Assuming the 2D array to be a square matrix with odd dimensions i.e., 3x3, 5x5, 7x7 eg.,
   4 5 6output through the function should be
   7 8 9Middle row 7 8 9
   3 2 1 Middle col. 5 8 2
5. WAP in C++ which will accept a 2 D Array of integer and return the sum of all the elements divisible by 5 that lie on the even row number.
6. WAP in C++ to print the product of each column of a two dimensional integer array passed as the argument of the function. Explain: if the two dimensional arrays contains

1	2	4
3	5	6
4	3	2
2	1	5

Then the output should appear as: Product of Column 1=24, Product of Column 2=30, Product of Column 3=240

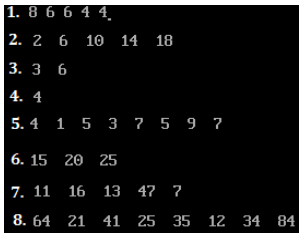
7. WAP in C++ which accepts a 2-D array of integers and its size as arguments and prints no of even numbers and odd numbers in each column.

If the array is

11	12	31	41
52	62	71	82
9	10	11	12

The output will be:

Column 1:	Even numbers : 1	Odd numbers : 2
Column 2:	Even numbers : 3	Odd numbers : 0
Column 3:	Even numbers : 0	Odd numbers : 3
Column 4:	Even numbers : 2	Odd numbers : 1



1 Define a function Reversearray(int[], int) that would accept a one dimensional integer array NUMBERS and its size N. The function should reverse the content of the array without using any second array.
Note: use the concept of swapping the elements.
Eg: if the array initially contains
2,15,7,8,10,1,13
after swapping should contain 31,1,10,8,7,15,2

Given an array named A with following elements
3,-5,1,3,7,0,-15,3,-7,-8

2 write a C++ function to shift all the negative numbers to left so that the resultant array may look like
-5,-15,-7,-8,3,1,3,7,0,3

3 Write a function in C++ which accepts an integer array and its size as arguments and swaps the elements of every even location with its odd location
eg.. if the array initially contains
2, 4, 1, 6, 5, 7, 9, 2, 3, 10
then it should contain
4, 2, 6, 1, 7, 5, 2, 9, 10, 3

4 From a 2D array ARR[3][3] write a program to prepare one dimensions array ARR2[9] that will have all the elements of ARR as if they are stored in row major form.
1 2 3
4 5 6
7 8 9
then should contain 123456789

5 Consider a 1D array A containing N integers. Develop an algorithm to do the following.
(i) Remove all occurrences of a given integer
(ii) Shift the elements of the array to the right so that unused space is available at the left end.
(iii) Fill the unused spaces with zero.

6 Arrange the following array of integers in ascending order using bubble sort technique.
Array elements are: 26, 21, 20, 23, 29, 17, 14

7 Write a function check( ) to check if the passed array of 10 integers is sorted or not. The function should return 1 if arranged in ascending order, -1 if arranged in descending order, 0 if it is not sorted.

8. Write a function in C++ which accepts an integer array and its size as arguments and replaces laments having even values with its half and elements having odd values with twice its value

9. Write a function in C++ which accepts an integer array and its size as argument and exchanges the value of first half side elements with the second half side elements of the array.
Example: If an array of eight elements has initial content as 2,4,1,6,7,9,23,10. The function should rearrange the array as 7,9,23,10,2,4,1,6.

10. Write a function in c++ to find and display the sum of each row and each column of 2 dimension array. Use the array and its size as parameters with int as the data type of the array.

11. Write a function in C++, which accepts an integer array and its size as parameters and rearrange the array in reverse. Example if an array of five members initially contains the elements as 6,7,8,13,9,19 Then the function should rearrange the array as 19,9,13,8,7,6

12. Write a function in C++, which accept an integer array and its size as arguments and swap the elements of every even location with its following odd location. Example : if an array of nine elements initially contains the elements as 2,4,1,6,5,7,9,23,10 Then the function should rearrange the array as 4,2,6,1,7,5,23,9,10

13. Write a function in C++ which accepts an integer array and its size as arguments and replaces elements having odd values with thrice and elements having even values with twice its value.
Example: If an array of five elements initially contains the elements 3,4,5,16,9 Then the function should rearrange the content of the array as 9,8,15,32,27

Set 5
Array using Function
(2 or 3 Marks) Solve

1. **Write definition for a function DISPMID(int A[][5],int R,int C) in C++ to display the elements of middle row and middle column from a two dimensional array A having R number of rows and C number of columns.**

215	912	516	401	515
103	901	921	802	601
285	209	609	360	172

For example, if the content of array is as follows:
The function should display the following as output: 103 901 921 802 601, 516 921 609
Ans

```
void DISPMID(int A[][5],int R,int C)
{
for (int J=0;J<C;J++)
cout<<A[R/2][J]<< " ";
cout<<endl;
for (int I=0;I<R;I++)
cout<<A[I][C/2]<< " ";
}
```

OR

```
void DISPMID(int A[][5],int R,int C)
{
if(R%2!=0)
{
for (int J=0;J<C;J++)
cout<<A[R/2][J]<< " ";
}
else
cout<<"No Middle Row";
cout<<endl;
if(C%2!=0)
{
for (int I=0;I<R;I++)
cout<<A[I][C/2]<< " ";
}
else
cout<<"No Middle Column";
}
```

2. **Write a function REVROW(int P[][5],int N, int M) in C++ to display the content of a two dimensional array, with each row content in reverse order.**

For example, if the content of array is as follows:

15	12	56	45	51
13	91	92	87	63
11	23	61	46	81

The function should display output as:
51 45 56 12 15
63 87 92 91 13
81 46 61 23 81
Ans.

```
void REVROW(int P[][5],int N,int M)
{
for(int I=0; I<N; I++)
{ for(int J=M1;
J>=0; J)
cout<<P[I][J];
cout<<endl;
}
}
```

OR

```
void REVROW(int P[ ][5],int N,int M)
{
for(int I=0; I<N; I++)
{
for(int J=0; J<M/2; J++)
{
int T = P[I][J];
P[I][J] = P[I][MJ1]
;
P[I][MJ1]
= T;
}
}
for(I=0; I<N; I++)
{
for(int J=0; J<M; J++)
cout<<P[I][J];
cout<<endl;
}
}
```

3. **Write user-defined function AddEnd2(int A[ ][4], int N, int M) in C++ to find and display the sum of all the values which are ending with 2(ie units place is 2). For example if the content.**

22	16	12
19	5	2

The output should be 36
A

```
void AddEnd(int A[ ][4], int N, int M)
{
int I,j,Sum=0;
for(i=0;i<N;i++)
{
for(j=0;j<M;j++)
{
if(A[i][j]%10==2)
Sum=Sum+A[i][j];
}
}
cout<<Sum;
}
```

Write a user defined function DispTen(int A[ ][4], int N,int M) in C++ to find and display all the numbers which are divisible by 10. For example, if the content of array is :

12 20 13
2 10 30

The output should be 20 10 30

Answer)

```
Void DispTen(int A[ ][3], int N,int M)
{
int I,j,S=0;
for(i=0;i<N;i++)
for(j=0;j<M;j++)
if(A[i][j]%10= =0)
cout<<A[i][j]<<" ";
}
```

4. **Write a function ALTERNATE (int A[ ] [3], int N, int M) in C++ to display all alternate elements from two-dimensional array A (staring from A [0] [0]).** For example: **If the array is containing:**

23 54 76
37 19 28
62 13 19

The output will be23 76 19 62 19

Ans.

```
void ALTERNATE (int A [] [3], int N, int M)
{
int T=0;
for (int I=0 ; I<N; I++)
for (int J=0 ; J<M ; J++)
{
if (T%2= =0)
cout<<A[I] [J]<<" ";
T++ ;
}
}
OR
void ALTERNATE (int A[] [3], int N, int M)
{
int *P=&A[O] [0] ;
for (int I=0; I<N*M ; I+=2)
{
cout<<*p<<" ";
P+=2 ;
}
}
```

5. **Write a DSUMO function in C++ to find sum of Diagonal Elements from a NxN Matrix. (Assuming that the N is a odd number)**

Ans

```
void DSUM (int A [ ] [100] ,int N)
{
int SUMR =0, SUML=0;
for (int i=0; i<N;i++}
{
SUMR=SUMR + A[i] [i] ;
SUML = SUML + A[i] [N-1-i] ;
}
cout<< " Sum of Diagonal Elements =
"<<SUMR + SUML –
A[N/2] [N/2] ;
}
OR
void DSUM (int A[] [100], int N)
{
int SUMR =0, SUML=0;
for (int i=0; i<N; i++)
{
SUMR = SUMR + A[i] [i] ;
SUML = SUML + A[i] [N-1-i] ;
}
cout<< "Sum of Right Diagonal Elements =
"<<SUMR<<end1;
cout<< "Sum of Left Diagonal Elements =
"<<SUML<<end1;
}
OR
void DSUM (int A[] [100] , int N)
{
int SUMR =0, SUML=0;
for (int i = 0; i<N; i++)
{
for (int j = 0; j<N; j++)
{
if (i==j)
SUMR=SUMR + A[i] [j] ;
else if (i+j == N-1)
SUML = SUML + A[i] [j];
}
}
cout<< "Sum of Diagonal Elements ="
<< SUMR + SUML - A[N/2] [N/2];
}
```

6. **Write a function int SKIPSUM(int A[ ] [3], int N,int M) in C++ to find and return the sum of elements from all alternate elements of a two-dimensional array starting from A[0][0].**

Hint: If the following is the content of the array

A[0][0]	A[0][1]	A[0][2]
4	5	1
A[1][0]	A[1][1]	A[1][2]
2	8	7
A[2][0]	A[2][1]	A[2][2]
9	6	3

The function SKIPSUM() should add elements A[0][0],A[0][2], A[1][l],A[2][0] and A[2][2].

Ans)

```
int SKIPSUM(int A[ ][ 3 ], int N,int M)
{ int S=0:
for(int I = 0; 1< N; 1++)
for (int J = (I%2)?1:0; J<M; J = J+2)
S = S + A [I][J];
return S;
}
OR
int SKIPSlJM(int A[ ][3], int N, int M)
{ int S=0;
for (int I = 0; 1< N; I++)
for (int J = (I%2==0)? 0:1 ; J<M; J = J+2)
S = S + A [I][J];
return S;
}
OR
int SKIPSUM(int A[][3], int N, int M)
{
int I,J, S=O;
for (I = 0; 1 < N; 1++)
{
if (I%2)
//OR (1%2 !=0 ) OR (I%2 == 1)
J = l;
else
J = 0;
for ( ; J<M; J = J+2)
S = S + A[I][J];
}
return S;
}
OR
int SKIPSUM(int A[][3], int N, int M)
{
int S=0, C=0;
for(int I = 0; 1< N; 1++)
for (int J = 0; J<M; J++ )
{
if (C%2 == 0)
S = S + A[I][J];
C++;
}
return S;
}
OR
```

```
int SKIPSUM(int A[][3], int N, int M)
{
int S=0, C=l;
for(int I = 0; I<N; I++)
for (int J = 0; J<M; J++ )
{
if (C%2 != 0)
S = S + A[I][J];
C++;
}
return S;
}
```

OR

```
int SKIPSUM (int A[ ][3], int N, int M)
{
int S=0;
for(int I = 0; l< N; l++)
for (int J = 0; J<M; J++ )
{
if ((I+J)%2 == 0)
S = S + A[I][J];
}
return S;
}
```

7. Write a function int ALTERSUM (int B[ ] [5], int N, int M in C++ to find and return the sum of elements from all alternate elements of a two-dimensional array starting from B[0][0]. Hint: If the following is the content of the array:

B[0][0]	B[0][1]	B[0][2]
4	5	1
B[1][0]	B[1][1]	B[1][2]
2	8	7
B[2][0]	B[2][1]	B[2][2]
9	6	3

The function should add elements B[0][0], B[0][2], B[l][l],B[2][0] and B[2][2].

Ans.

```
int ALTERSUM(int B[ ][5] ,int N,int M)
{
int Sum=0;
for (int I=0;I<N;I++)
for (int J=(I%2==0)?0:1;J<M;J+=2)
Sum+=B[I][J] ;
return Sum;
}
```

OR

```
int ALTERSUM(int B[ ][5],int N,int M)
{
int Sum=0,J=0;
for (int I=0;I<N;I++)
{
for (;J<M;J+=2)
Sum+=B[I][J] ;
J-=M;
}
return Sum;
}
```

OR

```
int ALTERSUM(int B[ ][5],int N,int M)
{
int *P=&B[0][0],Sum=0;
for (int I=0;I<M*N;I+=2)
{
Sum+=(*P);
P+=2;
}
return Sum;
}
```

OR

```
int ALTERSUM (int B[ ][5], int N, int M)
{ int S=0, C=0;
for(int I = 0; 1< N; 1++)
for (int J = 0; J<M; J++ )
{
if (C%2 == 0)
S = S + B[I][J];
C++;
}
return S;
}
```

OR

```
int ALTERSUM(int B[ ][5],int N,int M)
{ int Sum=0;
for (int I=0;1<N;1++)
for (int J=0;J<M;J++)
if ((I+J)%2==0)
Sum+=B [I][J] ;
return Sum;
}
```

OR

```
int ALTERSUM(int B[ ][5],int N,int M)
{
int Sum=0;
for (int I=0;1<N;1++)
for (int J=0;J<M;J++)
{
if ((I%2==0 && J%2=0)||(1%2!=0
&& J%2!=0))
Sum+=B [I][J] ;
}
return Sum;
}
```

8. Write a function in C++ to print the product of each column of a two dimensional array passed as the arguments of the function.

1	2	4
3	5	6
4	3	2
2	1	5

Example : If the two dimensional array containsThen the output should appear as:Product of Column 1 = 24Product of Column 2 = 30Product of Column 3 =240

A.

```
void receive(int A[ ][ ],int r,int c)
{ int i,j,B[c];
for(i=0;i<c;i++)
B[i]=1;
for(i=0;i<r;i++)
for(j=0;j<c;j++)
B[j]=B[j]*A[i][j];
for(i=0;i<c;i++)
cout<<"\nProduct of Column
"<<i+1<<" = "<<B[i];
}
```

OR

```
void ProdCol(int Arr[][100], int Row, int
Col)
{
int i, j, Prod;
for (j = 0; j < Col; j++)
{
Prod=1;
for (i = 0; i < Row; i++)
Prod * = Arr[i][j];
cout<<"Product of Column"
<<j<< "=" <<Prod<<endl;}}
```

9. Write a function in C++ to print the product of each row of a two dimensional array passed as the arguments of the function

20	40	10
40	50	30
60	30	20
40	20	30

Example: if the two imensional array contains
Then the output should appear as:
Product of Row 1 = 8000
Product of Row 2 = 6000
Product of Row 3 =3600
Product of Row 4 = 2400

```
void receive(int A[ ][ ],int r,int c)
{ int i,j,B[r];
for(i=0;i<r;i++)
B[i]=1;
for(i=0;i<r;i++)
for(j=0;j<c;j++)
B[i]=B[i]*A[i][j];
for(i=0;i<r;i++)
cout<<"\nProduct of Row "<<i+1<<
" = "<<B[i];
}
```

10. Write a function in C++ which accepts a 2D array of integers and its size as arguments and displays the elements which lie on diagonals. [Assuming the 2D Array to be a square matrix with odd dimension i.e., 3x3, 5x5 ,7x7 etc…]

Example : if the array content is
5 4 **3**
6 **7** 8
1 2 **9**
Out put through the function should be : Diagonal One : 5 7 9, Diagonal Two : 3 7 1

Solution:

```
void accept(int a[ ][ ],int size)
{ int i,j;
cout<<"Diagonal One:";
for (int i=0;i<size;i++)
for(int j=0;j<size;j++)
if (i= = j)
cout<<a[i][j]<<'\t';
cout<<"\n Diagonal Two:";
for (i=0;i<size;i++)
for(j=0;j<size;j++)
if((i+j)= =(size-1))
cout<<a[i][j]<<'\t';
}
```

11. Write a function in C++ which accepts a 2D array of integers and its size as arguments and displays the elements of middle row and the elements of middle column. [Assuming the 2D Array to be a square matrix with odd dimension i.e., 3x3, 5x5, 7x7 etc…]

Example : If the array content is
3 5 4
7 6 9
2 1 8
Output through the function should be :Middle Row : 7 6 9 Middle Column : 5 6 1

Solution:

```
void accept(int a[ ][ ],int size)
{
int i,j;
cout<<"Middle Row:";
for (int i=0;i<size;i++)
for(int j=0;j<size;j++)
if (i= = size/2)
cout<<a[i][j]<<'\t';
cout<<"\n Middle Column:";
for (i=0;i<size;i++)
for(j=0;j<size;j++)
if(j= =size/2)
cout<<a[i][j]<<'\t';
}
```

12. Write a function in C++ to print sum of all values which either are divisible by 2 or divisible by 3 present in a 2D array passed as the argument of the function.

Solution:

```
void Sum(int A[ ][ ],int R,int C)
{
int i,j,S=0;
for(i=0;i<R;i++)
for(j=0;j<C;j++)
if(A[i][j]%2= = 0 ||A[i][j]%3= = 0)
S=S+A[i][j];
cout<<"\nThe Sum of all the values which
are divisible by 2
or 3 in the array = "<<S;
}
```

13. Write a function in C++ to print sum of all values which either are divisible by 3 or divisible by 5 present in a 2D array passed as the argument of the function.

Ans)

```
void Sum(int A[ ][ ],int R,int C)
{ int S=0,i,j;
for(i=0;i<R;i++)
for(j=0;j<C;j++)
if((a[i][j]%3= =0)||(a[i][j]%5= =0))
S=S+A[i][j];
cout<<" nThe Sum of all the values
which are divisible by 3 or 5 in
the array = "<<S;
}
```

14. Write a function in C++ to find the sum of diagonal elements from a 2D array of type float. Use the array and its size as parameters with float as its return type.

Solution:

```
float diasum(float A[ ][ ],int R,int C)
{
int i,j;
float Dsum=0.0;
for(i=0;i<R;i++)
for(j=0;j<C;j++)
if((i= = j)| | (i+j)= =(size-1))
Dsum=Dsum+A[i][j];
return Dsum;
}
```

15. Write a user-defined function in C++ to display those elements of 2D array T[4][4] which are divisible by 100. Assume the content of the array is already present and the function prototype is as follows: void showhundred(int T[4][4]);

```
void showhundred(int T[4][4])
{
int i,j;
cout<<"\nThe elements in the array
which are divisible by 100 …..";
for(i=0;i<4;i++)
for(j=0;j<4;j++)
if(T[i][j]%100= =0)
cout<<T[i][j]<<'\t';
}
```

16. Write a user-defined function named Lower_half() which takes 2D array A, with size N rows and N columns as argument and prints the lower half of the array.

Eg:

Input:

2 3 1 5 0
7 1 5 3 1
2 5 7 8 1
0 1 5 0 1
3 4 9 1 5

Output:

2
7 1
2 5 7
0 1 5 0

```
3 4 9 1 5
```

Solution:

```
void Lower_half( int A[ ][ ],int N)
{ int i,j;
for(i=0;i<N;i++)
for(j=0;j<N;j++)
{ if(i<j)
cout<<A[i][j]<<'\t';
cout<<endl;}}
```

17. Write a user-defined function in C++ to find and display the multiplication of row elements of two dimensional array A[4][6] containing integers.

```
void rowmul( )
{ int A[4][6],i,j,rowmul;
cout<<"\nEnter any 24 values…";
for(i=0;i<4;i++)
for(j=0;j<6;j++)
cin>>A[i][j];
for(i=0;i<4;i++)
{
rowmul=1;
for(j=0;j<6;j++)
rowmul=rowmul*A[i][j];
cout<<"\nThe multiplication of
"<<i+1<<" row = "<<rowmul;}}
```

18. Write a user-defined function in C++ to find and display the sum of diagonal elements from a 2D array R[7][7] containing integers.

```
void displaysum( )
{ int i,j,D1=0,D2=0,R[7][7];
cout<<"\nEnter any 49 values….";
for(i=0;i<7;i++)
for(j=0;j<7;j++)
{
cin>>R[i][j];
if(i= = j)
D1=D1+R[i][j];
else if ((i+j)= =(size-1))
D2=D2+R[i][j];
}
cout<<"\nThe sum of the elements of
the Main Diagonal = "<<D1;
cout<<"\nThe sum of the elements of
the Other Diagonal = "<<D2;}
```

19. Write a function in C++ to find the sum of both left and right diagonal elements from a two dimensional array (matrix).

Ans)

```
void DiagSum(int M[][4],int N,int M)
{ int SumD1=0,SumD2=0;
for (int I=0;I<N;I++)
{
SumD1+=M[I][I];SumD2+=M[N-I-1][I];
}
cout<<"Sum of Diagonal 1:"
<<SumD1<<endl;
cout<<"Sum of Diagonal 2:"
<<SumD2<<endl;}
```

20. Write a function in C++ to find sum of rows from a two dimensional array.

Ans)

```
void MatAdd(int M[][4],int N,int M)
{
for (int R=0;R<N;R++)
{
int SumR=0;
for (int C=0;C<M;C++)
SumR+=M[C][R];
cout<<SumR<<endl;}}
```

h)Write a function in C++ to find the sum of both left and right diagonal elements from a two dimensional array (matrix).

Ans)

```
void DiagSum(int A[100][100],int N)
{
int SumD1=0,SumD2=0;
for (int I=0;I<N;I++)
{
SumD1+=A[I][I];SumD2+=A[N-I-1][I];
}
cout<<"Sum of Diagonal 1:"
<<SumD1<<endl;
cout<<"Sum of Diagonal 2:"
<<SumD2<<endl;}
```

21. Write a function in C++ to find sum of rows from a two dimensional array.

Ans)

```
void MatAdd(int A[100][100], int N,int M)
{
for (int R=0;R<N;R++)
{
int SumR=0;
for (int C=0;C<M;C++)
SumR+=A[C][R];
cout<<SumR<<endl;}}
```

Array Base Address

Row Major: A[I][J] = B+W[N(I-LR) + (J-LC)]

Column Major: A[I][J] = B+W[M(J-LC) +(I-LR)]

Where A[i][j]= the element whose address is to be found, B = base address of array ,W = size of each element

Lr = lowest row index, Lc = lowest column index, M = total number of rows in array, N= total number of columns in array

Base Address Calculation of 2-D array. (Row-Major) (3 Marks)[Solve]

1. R[10][50] is a two dimensional array, which is stored in the memory along the row with each of its element occupying 8 bytes, find the address of the element R[5][15], if the elementR[8][10] is stored at the memory location 45000.

A)Loc(R[I][J])=BaseAddress + W [(I – LBR)*C + (J – LBC)]

(whereW=size of each element = 8 bytes, R=Number of Rows=10, C=Number of Columns=50)
Assuming LBR = LBC = 0

LOC(R[8][10])
45000 = BaseAddress + W[I*C + J]
45000 = BaseAddress + 8[8*50 + 10]
45000 = BaseAddress + 8[400 + 10]

45000 = BaseAddress + 8 x 410
BaseAddress = 45000 3280= 41720
LOC(R[5][15])= BaseAddress + W[I*C + J]
= 41720 + 8[5*50 + 15]
= 41720 + 8[250 + 15]
= 41720 + 8 x 265
= 41720 + 2120
= 43840

OR

Loc(R[I][J])
=Reference Address + W [(I – LR)*C + (J – LC)]
(where W=size of each element = 8 bytes, R=Number of Rows=10, C=Number of Columns=50)
Reference Address= Address of given cell R[8][10]=45000
LR = Row value of given cell = 8
LC = Column value of given cell = 10
LOC(R[5][15])= LOC(T[8][10]) + 8[(5 8)*50 + (15 10)]
LOC(R[15][5]) = 45000 + 8[3*50 + 5]
= 45000 + 8[150+ 5]
= 45000 + 8 x (145)
= 45000 1160
= 43840

2. A two dimensional array ARR[50][20] is stored in the memory along the row with each of its elements occupying 4 bytes. Find the address of the element ARR[30][10], if the element ARR[10] [5] is stored at the memory location 15000.

Loc(ARR[I][J]) along the row
=BaseAddress + W [(I – LBR)*C + (J – LBC)]
(where C is the number of columns, LBR = LBC = 0
LOC(ARR[10][5])
= BaseAddress + W [I*C + J] 15000 = BaseAddress + 4[10*20 + 5]
= BaseAddress + 4[200 + 5]
= BaseAddress + 4 x 205
= BaseAddress + 820
BaseAddress = 15000820
= 14180
LOC(ARR[30][10])= 14180 + 4[30 * 20 + 10]
= 14180 + 4 * 610
= 14180 + 2440
= 16620

OR

LOC(ARR[30][10])
= LOC(ARR[10][5])+ W[(ILBR)*C + (JLBC)]
= 15000 + 4[(3010)*20 + (105)]
= 15000 + 4[20*20 + 5]
= 15000 + 4 *405
= 15000 + 1620
= 16620

OR

Where C is the number of columns and LBR=LBC=1
LOC(ARR[10][5])
15000 = BaseAddress + W [(I1)*C + (J1)]
= BaseAddress + 4[9*20 + 4]
= BaseAddress + 4[180 + 4]
= BaseAddress + 4 * 184
= BaseAddress + 736
BaseAddress = 15000 736
= 14264
LOC(ARR[30][10])= 14264 + 4[(301)*20 + (101)]
= 14264 + 4[29*20 + 9]
= 14264 + 4[580 + 9]
= 14264 + 4*589
= 14264 + 2356
= 16620

3. An array A[20][30] is stored along the row in the memory with each element requiring 4 bytes of storage. If the base address of array A is 32000, find out the location of A[15][10].Also find the total number of elements present in this array.

Answer)
B=32000 W=4
A[15][10]=32000+4[30(15-0)+(10-0)]
=32000 +4[450+10]
=32000+4[460]
=32000+1840
=33840
Location of a[10][15]=33840
Total number of elements present in this array = 20*30 = 600

4. An array T[15][10] is stored along the row in the memory with each element requiring 8 bytes of storage. If the base address of array T is 14000, find out the location of T[10][7];

Answer)
Address of T[10][7]=14000+(10*7+10)*8
=14000+(80)*8
=14000+640
=14640

5. An array G[50][20] is stored in the memory along the row with each of its elements occupying 8 bytes. Find out the location of G[10][15], if G[0][0] is stored at 4200.

(2011 OD) 3
Ans
Assuming LBR=LBC=0
B=4200
W=8 bytes
Number of Rows(N)=50
Number of Columns (M)=20
LOC(Arr[I] [J]) = B +(I*M + J) *W
LOC (Arr [10] [15]) = 4200+ (10*20+15)*8
= 4200 + (215*8)
= 4200+1720
= 5920

6. An array Arr[50][10] is store in the memory along the row with each element occupying 2 bytes. Find out the Base address of the location Arr[20][50], if the location Arr[10][25]is stored at the address 10000.

Ans)
Assuming LBR=LBC=0
S=2 bytes
Number of Rows (N)=50
Number of Columns (M)=100
LOC (Arr [I] [J]) = B + (I*M+J)*S
LOC (Arr [10] [25]) = B +(10*100+25)*2
10000 = B +(1000+25)*2
B = 10000-2050
B = 7950
LOC (Arr [20] [50])
= 7950+(20*100+50)*2
= 7950 + (2050*2)
= 7950+4100
= 12050

OR

Assuming LBR=LBC=1
S=2 bytes
Number of Rows (N) = 50
Number of Columns (M) =100
LOC (Arr [I] [J]) =B +((I-LBR)*M+(J-LBC))*S
LOC (Arr [10] [25]) =B +((10–1)*100+(25–1))*2
10000 = B +(900+24)*2
B = 10000-1848
B = 8152
LOC (Arr [20] [50])
= 8152+ ((20-1)*100+ (50-1))*2
= 8152 + (1949*2)
= 8152+3898
=12050

7. An array Arr[15][20] is stored in the memory along the row with each element occupying 4 bytes. Find out the Base address of the location Arr[3][2], if the location Arr[5][2] isstored at the address 1500.

Solution:
Given Data: Arr[15][20] W=4 B=? R=15 C=20 Lr= 0 Lc = 0
Address of Arr[3][2] = ?
Address of Arr[5][2] = 1500.
Address of an element (I,J) in row major = B+W(C(I-Lr)+(JLc))
Therefore,
1500 = B+4(20(5-0)+(2-0))
1500 = B+4(20*5+2)
1500 = B+4*102
1500 = B+408
B =1500-408
B=1092
Address of Arr[3][2]

=1092+4(20*3+2)
=1092+4(62)
=1092+248
=1340.

8. An array MAT[20][10] is stored in the memory along the row with each element occupying 4 bytes of the memory. Find out the Base address and the address of element MAT[10][5] ,if the location MAT[3][7] is stored at the address 1000.

Ans) For Row wise allocation
Address of A[I][J]= BA + W((I-LBR) x N + (J-LBC))
WhereBA = Base Address
W = Size of each element in bytes = 4 bytes (given)
N = No. of columns in the 2D Array = 10 (given)
Address of MAT[3][7] given is 1000.Therefore
(Assumption 1: LBR = LBC = 0)
MAT[3][7]=100 = BA + 4 (10 (3-0) + (7-0))
= BA + 148
BA = 1000 – 148 = 852
Therefore,Base Address = 852
Thus, Address of MAT[10][5] = 852 + 4 (10 (10-0) + (5-0))
= 852+420
= 1272
OR
(Assumption 2: LBR = LBC = 1)
MAT[3][7]=1000 = BA + 4 (10 (3-1) + (7-1))
= BA + 104
BA = 1000 – 104
= 896
Therefore,Base Address = 896
Thus, Address of MAT[10][5]
= 896 + 4 (10 (10-1) + (5-1))
= 896+376
= 1272

9. An array Arr[15][35] is stored in the memory along the row with each of its element occupying 4 bytes . Find out the Base address and the address of element Arr[2][5] , if thelocation Arr[5][10] is stored at the address 4000.

Ans) LOC(Arr[I][J])=Base(Arr)+W*(I + No.of Rows * J)
LOC(Arr[5][10])=Base(Arr)+8*(5+15*10)
4000 =Base(Arr)+8*(155)
4000 =Base(Arr)+1240
Base(Arr) =4000-1240
Base(Arr) =2760
LOC(Arr[2][5]) =Base(Arr)+8* (2 + 15*5)
=2760+8*(77)
=2760+616
=3376

OR
LOC(Arr[I][J])
=Base(Arr)+W*((I-1) + No. of Rows * (J-l))
LOC(Arr[5][10])
=Base(Arr)+8*[(5-1)+15* (10-1)]
4000 =Base(Arr)+8*(139)
4000 =Base(Arr)+1112
Base(Arr) =4000-1112
Base(Arr) =2888
LOC(Arr[2][5])
=Base(Arr)+ 8*[(2-1)+15*(5-1)]
=2888+8*(61)
=2888+488
=3376

10. An array Arr[35][15] is stored in the memory along the row with each of its element occupying 4 bytes . Find out the Base address and the address of element Arr[20][5] , if thelocation Arr[2][2] is stored at the address 3000.

Ans)
LOC(Arr[I][J])
Base(Arr)+W*(No. of Cols*I+J)
LOC(Arr[2][2]) =Base(Arr)+4*(15*2+2)
3000 =Base(Arr)+4*(32)
3000 =Base(Arr)+128
Base(Arr) =3000-128
Base(Arr) =2872
LOC(Arr[20][5])
=Base(Arr)+4*(15*20+5)
=2872+4*(300+5)
=2872+4*305
=2872+1220
=4092
OR
LOC(Arr[I][J])
=Base(Arr)+W*(No. of Cols*(I-1)+(J-1)
LOC(Arr[2][2])
=Base(Arr)+4*(15*(2-1)+(2-1))
3000 =Base(Arr)+4*(16)
3000 =Base(Arr)+64
Base(Arr) =3000-64
Base(Arr) =2936
LOC(Arr[20][5])
=Base(Arr)+4*(15*(20-1)+(5-1))
=2936+4*(289)
=2936+1156
=4092

11. An array S[40][30] is stored in the memory along the row with each of the element occupying 2 bytes, find out the memory location for the element S[20][10], if the BaseAddress of the array is 5000.

Ans)
Given,W=2
N=40
M=30
Base(S)=5000
Row Major Formula:
Loc(S[I][J])
=Base(S)+W*(M*I+J)
Loc(S[20][10])
=5000+2*(30*20+10)
=5000+2*(600+10)
=5000+1220
=6220

12. An array S[40][30] is stored in the memory along the row with each of the element occupying 2 bytes, find out the memory location for the element S[20][10], if an elementS[15][5] is stored at the memory location 5500.

Ans)
Given,
W=2
N=40
M=30
Loc(S[15][5])=5500
Row Major Formula:
Loc(S[I][J]) =Base(S)+W*(M*I+J)
Loc(S[15][5] =Base(S)+2*(30*15+5)
5500 =Base(S)+2*(450+5)
Base(S) =5500- 910
Base(S) =4590
Loc(S[20][10]) =4590+2*(30*20+10)
=4590+2*(600+10)
=4590+1220= 5810

13. An array T[15][10] is stored in the memory with each element requiring 2 bytes of storage. If the base address of T is 2000, determine the location of T[7][8] when the array VAL is stored (i) Row major (ii) Column major.

Unsolved Base Address Questions

1. An array VAL[1..15][1..10] is stored in the memory with each element requiring 4 bytes of storage. If the base address of array VAL is 1500, determine the location of VAL [12][9] when the array VAL is stored (i) Raw wise (ii)Column wise.**[1972/ 2024]**

2. An array DAYA [1..10][1..10] requires 8 bytes of storage. If the base address of array DATA is 1500,determine the location of DATA [4][5], when the array DATA is stored(i) Raw wise (ii)Column wise. **[1772/1844]**

3. X[1..16][1..10] is a two -dimensional array. The first element of the array is stored at location 100. Each element of array occupies 6 bytes. Find the memory location of X[2][4] when (i) array is stored row wise and (ii) array is stored column wise.**[178/394]**

4. An array X[7][20] is stored in memory with each element requiring 2 bytes of storage. If the base address of the array is 2000, calculate the location of X[3][5] when the array X is stored in column major order.**[2130/2076]**

Note : X[7][20] means valid row indices are 0 to 6 and valid column indices are 0 to 19.

5. An array X[10][20] is stored in memory with each element requiring 4 bytes of storage. If the base address of the array is 1000, calculate the location of X[5][15] when the array X is stored using column major order.**[1460/1620]**

Note : X[10][20] means valid row indices are 0 to 9 and valid column indices are 0 to 19.

6. A two -dimensional array A[10][3] is stored in the memory. The first element of the array is stored at location 140. Find the memory location of A[5][2] if each element of the array requires 4 memory locations and the array is stored (i) Raw wise (ii)Column wise.**[208/240]**

7. The array A[20][10] is stored in the memory with each element requiring one Byte of storage if the base address of A is C0, determine the location of A[10][5] when the array A is stored by column major.**[C+110/C+105(RM)]**

8. An array V[15][30] is stored in the memory with each element requiring 8 bytes of storage. If the base address of V is 5300, find out memory locations of V[8][12] and [12][2],if the array is stored along the row.**[(CW)6804/(RW)7316//8196/5636]**

9. An array X[30][10] is stored in the memory with each element requiring 4 bytes of storage. If the base address of X is 4500, find out memory locations of X[12][8] and X[2][14], if the content is stored along the row.**[5012/4636]**

10. An array ARR [5][5] is stored in the memory with each element occupying 4 bytes of space. Assuming the base address of ARR to be 1000,compute the address of ARR [2][4], when the array is stored:(i) Raw wise (ii)Column wise.**[1056/1088]**

11. An array ARR [5][5] is stored in the memory with each element occupying 2 bytes of space. Assuming the base address of ARR to be 1500,compute the address of ARR [2][4], when the array is stored:(i) Raw wise (ii)Column wise.**[1528/1544]**

12. An array A [10][20] is stored in the memory with each element requiring 2 bytes of storage. If the base address of array in the memory is 800,determine the location of A[9][11] when the array is stored as(i)Row major,(ii)Column major.**[1182/1038]**

13. An array A[10][20] is stored in the memory along the column with each of the element occupying 2 bytes, find out the memory location for element A[2][5],if an element A[5][10] is stored at the memory location 3020.**[2810/2914]**

14. An array of real numbers RealArr[20][20], find the address of Real Arr[10][12] if RealArr[1][1] is stored in location 1000. Assume each real numbers require 4 bytes. Show steps in your calculation.

15. Given the array AAA[50] with base address 300 and element size 4 bytes. Find the address of AAA[10], AAA[25] and AAA[40].

16. For an array of real numbers Realarr[20][20], find the address of Realarr[10][12] if Realarr[1][1] is stored in location 1000. Assume each real number require 4 bytes. Show step in your calculation.

17. Each element of an array DATA [20][50] requires 4 bytes of storage. Base address of DATA is 2000, determine the location of DATA [10][10] when the array is stored as:(i)Row major,(ii)Column major.

18. A two dimensional array X[-2...5][3..7] is stored row- wise in the memory. The first element of the array is stored at location 100. Find the memory location of X[2][3] if each element of array requires 4 memory locations.

19. Given an array x[6][16] whose base address is 100.Calculate the location x[2][5] if each element occupies 4 bytes and array is stored row wise.

20. The array ARR[15][10] is stored in the memory with each element requiring one byte of storage if the base address of ARR is C0. Determine C0 when the location of ARR[11][6] is 1000.

21. An array Arr[15][35] is stored in the memory along the column with each of its element occupying 8 bytes. Find out the base address and the address of an element Arr[2][5],if the location Arr[5][10] is stored at the address 4000.

22. An array Arr[35][15] is stored in the memory along the row with each of its element occupying 4 bytes. Find out the base address and the address of an element Arr[20][5],if the location Arr[2][2] is stored at the address 3000.

23. An array MAT[30][10] is stored in the memory column wise with each element occupying 8 bytes of memory. Find out the base address and the address of element MAT[20][5], if the of MAT[5][7] is stored at the address 1000.

24. An array MAT[20][10] is stored in the memory row wise with each element occupying 4 bytes of memory. Find out the base address and the address of element MAT[10][5], if the of MAT[3][7] is stored at the address 1000.

25. An array Mat[15][7] is stored in the memory along the column with each element occupying 2 bytes of memory. Find out the base address and the address of element Mat[2][5], if the location of MAT[5][4] is stored at the address 100.

26. An array Array[20][15] is stored in the memory along the column with each element occupying 8 bytes. Find out the base address and the address of element Array[2][5], if the element Array[5][4] is stored at the address 1000.

27. An array Arr[15][20] is stored in the memory along the row with each element occupying 4 bytes. Find out the base Address and address of the element Arr[3][2] if the element Array [5][2] is stored at the address 1500.

28. An array Arr[15][20] is stored in the memory along the row with each element occupying 4 bytes. Find out the base Address and address of the element Arr[3][2] if the element Arr[5][2] is stored at the address 1500.

29. An array Arr[40][10] is stored in the memory along the column with each element occupying 4 bytes. Find out the base Address and address of the location Arr[3][2] if the location Arr[5][2] is stored at the address 9000.

answer

(1)given base=1500, W=4bytes , N=10, M=15 , I=12 ,and J= 9,(i) raw major order,The formula is applied :

VAL[I][J]= B+((I-1)*N+(J-1))*W

=1500+((12-1*10+(9-1))*4

=1500+(110+8)*4

=1500+118*4

=1972

(2) given base=1500,W=8bytes ,N=10,M=10,I=4 and J=5

(i)row major order
The formula is aaplied
VAL[I][J]=B+((I-1)*N+(J-1))*W
= 1500+((4-1)*10+(5-1))*8
=1500+(34)*8
=1500=272
=1772

(ii) coloum major order
the formula is applied
VAL[I][J]=B=((J-1)*M+(I-1)*W
=1500+((4-1) +10*(5-1)*8
=1500+(43)*8
=1500+344
=1844

(3) given B =100,W =6, N = 10 , M =16

(i) row major order
The formula is applied :
X [I][J] = B+w (n(I-L1)+ (J-L2))
X [2][4] = 100+6*(10*(2-1) + (4-1)
= 100+6*(10+3)
= 100+6*13
=100=78
=17

(ii) coloum major order
the formula is applied :
x[I][J] =B+w((I=L1) +m(J-L2))
X[2][4] =100+6*((2-1) + 16*(4-1))
= 100+6*49
= 100 +294
= 394

(5) The location is as
X [5][5] =B+W((I-L1)+m(J-L2))
=1000+4((5-0) +10 (15-0)
= 1000+4((5-150)
= 1000 + 620
= 1620

(7) A[I][J] = B+W((I-L1)+M(J-L2)) B= C W = 1
A[10][5] = C+1((10-0)+20(5-0))
= C + 1 (10+20x 5)
=c +1 (10+100)
= c + 110

(8) the memory location is as:
V[I][J] = B+W(N(I-L1)+(J-L2))
V[8][12] =5300+8(30(8-0)+(12-0)
= 5300=8(240+12)
= 5300+8 x252
= 5300+2016
= 7316

V[12][2] = 5300+8(30(12-2)+(2-0)
= 5300+8(360+2)
= 5300+8 x 362
= 5300+2896
= 8196

(9) The memory location is as :
X[I][J] = B+W(N(I-L1)+ (J-L2))
X[12][8] = 4500+4(10(12-0)+(8-0)
= 4500+4(120+8)
= 4500+4 x 128
= 4500+512
= 5012

X[2][14] = 4500+4(10(2-0)+(14-0)
= 4500+4(20+14)
= 4500+4 x 34
= 4500+136
= 4636

(10) Given Let us assume that the base index number is [0][0].
The base address is T = B =1000, No .of rows = R= 5, No . of Coloum =C = 5,Size of each element =W=4,
And lets the location is ARR[2][4] = L

(i)ARR is stored as row major
Using the formula
L = B +W*[2*C+4]
= 1000+4*[2*5+4]
= 1000+4*[10+4]
= 1000+4*14
= 1000+56
= 1056

(ii) ARR is stored as coloum major
using the formula
L = B + W*[2+4*R]
= 1000 +4*[2+4*5]
= 1000+4*[2+20]
= 1000+4*22
= 1000+88
= 1088

(11) Given Let us assume that the base index number is [0][0].
The base address of T = B = 1500, No , of rows =R= 5, No , of coloum =C=5, Size of each element = W= 2,
And let the location of ARR [2][4]=L

(i) ARR is stored as row major
Using the formula
L= B+W*[2*C+4]
=1500+2*[2*5+4]
= 1500+2*[10+4]
= 1500+28

(ii) ARR is stored as coloum major
using the formula
L=B+W*[2+4*R]
= 1500+2*[2+4*5]
= 1500+2*[2+20]
= 1500+44

= 1528

(12) Given , B= 800 ,W = 2

(i) row major order

N = 20

A[9][11] = 800 +2(20(9-0)+(11-0))

= 800+2(20(9)+11)

= 800 +2(180+11)

= 800+2(191)

= 800+382

= 1182

(ii) coloum major order

m = 10

A[9][11] = 800+2((9-0)+10(11-0))

= 800+2(9+110)

= 800+2(119)

= 800+238

=1038

(13) Given W=2,M=10,N-20

LocA[5][10]=3020

Coloum major formula

A[I][J] =B+W ((I-0)+M(J-0))

A[5][10] =B+2((5-0)+10(10-0))

3020 = B+2(5+10*10)

Base (A) =3020-210

Base (A) =2810

A[2][5] = 2810+2((2-0)+10(5-0))

= 2810+2(2+10*5)

= 2810+2*52

= 2810+104

= 2914

(14) As we know

A[I][J] = B+w(n(I-L1)+(J-L2)),Now , B=1000,W=4,L1=L2=1

A[10][12] = 1000+4(20(10-1)+(12-1))=1000+4(20*9+11)

= 1000+764

= 1764

(15) As we knowA[I]=B+W(I-L)

Now B = 300, W = 4,L = 0

AAA [10] = 300 + 4 (10-0)

= 300 +4 (10-0)

= 300+40

= 340

AAA[25] = 300+4(25-0)

= 300+4(25)

= 300+ 100 =400

AAA[40]= 300+4 (40-0)

= 300+4(40)

= 300+160

= 460

(16) As we know

A[I][J] = B +w(n(I-L1)+(J-L2))

Now , B = 1000 w = 4, L1=L2=1

A[10][12]=1000+4*(20*(10-1)+912-1))

=1000+4*(20*9+11)

=1000+4*(180+11)

=1000+4*(191)=1000+764=1764

(17) B= 2000 W= 4

(i) row major order n = 50

DATA [10][10]=2000+4*(50*(10-0)+(10-0))

= 2000+4*(50*10+10)

= 2000+4*(500+10)

= 2000+4*510

= 2000+2040 = 4040

(18) B = 100 W = 4

L1 = -2 and L2 =3

N = U2 – L2 +I, = 7 – 3 + 1 = 5

X[2][3]= B+W(n(I-L1)+(J-L2))

= 100+4*(5*(2-(-2))+(3-3))

= 100+4*(5(2+2)+(3-3))

=100+4*(5*4+0)

= 100+4*20 = 100+80=180

(19) B=100 W= 4, L1=0 and L2 = 0,N = 16

X[2][5]= B+W(n(I-L1)+(J-L2))

= 100+4*(16*(2-0)+(5-0)

= 100+4*(16*2+5)

= 100+4*(32+5)

= 100+4*37

= 100+148

= 248

(20) A[I][J]=B+W((I-L1)+M(J-L2))

B=C

A[11][16] = C+I((11-0)+15(6-0))

1000 =C +1(11+15*6)

1000 = C +101

C = 1000 -101=899

(21) it is not specified that the array element are stored in the row major order

A[I][J]=B+W(n(I-L1)+(J-L2))

Now , B =100 and W = 2

L1 = 5 and L2 = 1

N = 20

A[-3][15] = 100+2(20(-3-(-5)) +(15-1))

=100+2(20(2)+14)

= 100+2(40+14)

= 100+2(54)

= 100+108 = 208

(22) the base address is :

Arr [i][j] = Base (Arr)+W *(i+no. of Rows *J)

Arr [5][10] = base (Arr)+8*(5+15*10)

4000 = base (Arr) +8*(155)

4000 = base (Arr) + 1240

Base (Arr)= 4000-1240

Base (Arr) = 2760

The address of :

Arr [2][5] = base (Arr)+8*(2+15*5)

= 2760 +8*(77)

= 2760 + 616 = 3376

(22) the base address is :

LOC (Arr[i][j]= Base (Arr)+W *(N0. Of coloum *i+j)

LOC (Arr[2][2]= Base(Arr)+4*(15*2+2)

3000 = base (Arr)+4*(32)

3000 = base (Arr)+128

Base (Arr) = 3000 -128

Base(Arr) = 2872

The address of :

Arr [2][5] = base (Arr)+8*(2+15*5)

= 2760 +8 *(77)

= 2760 + 616

= 3376

(11) the base address is :

LOC (Arr[i][j] = base (Arr)+W*(no. of coloum * I +j)

LOC (Arr [2][2] = base (Arr)+4*(15*2+2)

3000 = base (Arr)+4*(32)

3000 = base (Arr)+128

Base(Arr) = 3000 – 128

Base (Arr) =2872

The address is :

LOC (Arr[20][5] = base (Arr)+4*(15*20+5)

= 2872 + 4 *305

= 2872 + 1220 = 4092

(23) for coloum wise allocation

Address of A [I][J] = BA +W(J-LBC)Xm+(I-LBR)

Where ,

BA = base address

W = size of each element in bytes = 8 bytes (given)

M = no . of row in the 2D array = 30 (given)

Address is MAT[5][7] given is 1000.

Therefore,

MAT [5][7] = base address (MAT)+8 x (7 x 30 +5)

1000 = BA =8 x 215

= BA +1720

So , BA = 1000-1720 = -720

Therefore, base address = -720

Thus, address of MAT [20][5]

= -720+8(5x30+20)

=-720 +8x170

=-720+1360 = 640

(24) for row wise allocation

Address of A [I][J] = BA +W[(J-LBC)+N*(I-LBR)]

Where , BA = base address

W= size of each element in bytes = 4bytes (given)

N = no . of coloum in the 2D array = 10 (given)

Address of the MAT [3][7] given is 1000

Therefore,

LOC (MAT[3][7]= base address (MAT)+4x(3x10+7)

1000 = B A +4 x 37 =148

S0 ,BA = 1000 -148 = 852

Therefore, base address = 852

Thus, address of MAT[10][5]

= 852 + 4 (10x 10+5)

= 852 + 4 x 105

= 852+420

= 1272

(25) for coloum wise allocation

Address of A[I][J] = BA +W[(J-LBC) x M +(I-LBR)]

Where , BA = Base address

W = size of each element in bytes = 2 bytes (given)

M = NO. of row in the 2D array = 15 (given)

Address 0f MAT [5][4] given is 100.

Therefore,

LOC (MAT[5][4] = base address (MAT)+2x (4x 15+5)

100 = BA +2 (15x 4+5)

= BA +2x 65

= BA =130

So, BA = 100-130

= -30

Therefore , base address = -30

Thus, address of MAT[2][5]

= -30+2(15x5+2)

= -30+2x77

= -30+154

= 124

(26) For column wise allocation

Address of A [I][J] = BA+W[(J-LBC)x M+(I-LBR)]

Where ,

BA = base address

W = size of each element on bytes = 8 (given)

M = no . of row in the 2D array = 20 (given)

address of array [4][5] = base address (array) + 8x (5x 20+4)

1000 = BA +8x 104

BA + 832

So BA = 1000 -832

= 168

Therefore , base address = 168

Thus , address of array [2][3]

= 168+8(3x 20+2)

= 168 +8 x 62

= 168 + 496 = 664

(27) for row wise allocation

Address of A [I][J] = BA+ W[(J-LBC)+Nx(I-LBR)]

Where , BA = base address

W = size of each element in bytes = 4 bytes (given)

N = no. of coloum in the 2D array = 20 (given)

Address of array [5][2] given is 1500.

Therefore,

Array[5][2] = base address (array) +4x (5x20+2)

1500 = BA +4x 102

= BA +408

So , BA = 1500 -408

= 1092

Therefore ,base addrerss = 1092

Thus , address of array [3][2]

= 1092 +4(3x 20 +2)

= 1092 + 4x 62

= 1092 +248 = 1340

(29) for coloum wise allocation

Address of A [I][J] = BA +W[(J-LBC)x m+(I-LBR)]

Where ,

BA = base address

W = size of each element in bytes = 4 bytes (given)
M = No . of row in the 2D array = 40 (given)
Address of array [30][10] given is 9000.
Therefore,
Array [30][10]) = base address (array) +4x (10x40+30)
9000= BA +4x 430
= BA +1720
So, BA = 9000 -1720
= 7280
Therefore, Base Address =7280
Thus, Address of Array[3][6]
=7280+4x (6x40+3)
= 7280 + 4 x 243
= 7280 + 972
= 8252

MODEL 3B: Address Calculation of 2-D array. (Column-Major) (3 Marks)

(b)An array P[30][20] is stored along the column in the memorywith each element requiring 2 bytes of storage. If the baseaddress of the array P is 26500, find out the location ofP[20][10]. (2016)3
Total number of rows= 30
Total size= 2 bytes
Base Address= 26500
LOC (P[I][J]) = BaseAddress+((I-LBR) + (J-LBC) * R)*W
Assuming Lower Bound of Row(LBR)=0
Lower Bound of Column(LBC)=0
Total number of Rows(R)=30
Size of each element(W)=2
LOC(P[20][10])= 26500 +((20-0)+(10-0)*30)*2
LOC(P[20][10])= 26500 +640
LOC(P[20][10])= 27140

(b) An array T[20][10] is stored in the memory along thecolumn with each of the elements occupying 2 bytes. Find outthe memory location of T[10][5], if the element T[2][9] isstored at the location 7600.
Ans
Assuming LBR=LBC=0
W=2 bytes
Number of Rows (M) =20

Number of Co1umns(N)=10
LOC(T[I] [J]) = B +(I + J*M)*W
LOC(T[2] [9]) = B +(2+9*20)* 2
XII Computer Chapter – 9 111
7600 = B + (182*2)
B = 7600 - 364
B = 7236
LOC (T[10] [5]) = 7236 +(10+5*20)* 2
= 7236 + (110*2)
= 7236 + 220
= 7456
OR
Assuming LBR=2, LBC=9 and B = 7600
W=2 bytes
Number of Rows (M) = 20
Number of Co1umns (N) = 10
LOC (T[I] [J]) = B + ((I-LBR) + (J-LBC)*M)*W
LOC (S[10] [5]) = 7600 + ((10-2) + (5-9)*20)*2
= 7600 + (8-80) * 2
= 7600 + (-72)) * 2
= 7600 - 144
= 7456
OR
Assuming LBR=LBC=l
W=2 bytes
Number of Rows (M) = 20
Number of Co1umns (N) = 10
LOC (T[I] [J]) = B +((I-LBR) + (J-LBC)*M)*W
LOC (T[2] [9]) = B +((2-1) + (9-1)*20)* 2
7600 = B + (161*2)
B = 7600 - 322
B = 7278
LOC (T[10] [5]) = 7278 +((10-1)+(5-1)*20)* 2
= 7278 + (9+80) *2)
= 7278 + 178
= 7456

(b) An array P[50] [60] is stored in the memory along thecolumn with each of the element occupying 2 bytes, find outthe memory location for the element P[10][20], if the BaseAddress of the array is 6800.

Ans) Loc(P[I] [J]) = Base(P)+W(I+J*M)i Loc(P[10][20]) = Base(P)+2(10+20*50)
Loc(P[10] [20]) = 68OO + 2(10+20*50)
= 6800 + 2 (10+1000)
= 6800 + 2*1010
= 6800 + 2020
= 8820
OR
Address of P[i] [j] = BaseAddress
+ W((i–L1)+(j–L2)*M)
Address of P[10] [20]= 6800 +
2((10-0)+(20-0)x50)
= 6800 + 2 x 1010
= 6800 + 2020
= 8820
OR
Address of P[I] [J] along the column
= BaseAddress + W((I–LBR)+(J–
LBC)*M)
(where N is the number of rows, LBR = LBC = 1)
Address of P[10][20]
=6800+2((10-1)+(20-l)x50)
= 6800 + 2 (9 + 19 x 50)
= 6800 + 2 x 959 = 6800 + 1918 = 8718

(b) An array T[90][100] is stored in the memory along the column with each of the elements occupying 4 bytes. Find out the memory location for the element T[10][40], if the Base Address of the array is 7200.

Ans. Loc(T[I][J)) = Base(T)+W(I+J*N)
(where N is the number of rows, LBR = LBC = 0)
= 7200 + 4[10 + 40 x 90]
= 7200 + 4[10+3600]
= 7200 + 4 x 3610
= 7200 + 14440
= 21640
OR
Address of T[I][J] along the column
= BaseAddress + W [(I-LBR)+(J-LBC)* N]
(where N is the number of rows, LBR=LBC = 1)
Address of T[10][40] = BaseAddress +
4[(10 - 1) +(40 - l)x 90]
= 7200+4[9 + 39 x 90]
= 7200+4[9 + 3510]
= 7200+4 x 3519
= 7200+14076
= 21276

(b) An array S[40][30] is stored in the memory along thecolumn with each of the element occupying 4 bytes, find outthe base address and address of element S[20][15], if anelement S[15][10] is stored at the memory location 7200.

Ans)Loc(S[I][J]) = Base(S)+W(I+J*N)
Loc(S[15][10]) =
Base(S)+4(15+10*40)
Base(S) = 7200-4*415
Base(S) = 7200-1660
Base(S) = 5540
Loc(S[20][15]) =
Base(S)+4(20+15*40)
Loc(S[20][15])
= 5540 + 4(20+15*40)
= 5540 + 4(20+600)
= 5540 + 4*620
= 5540 + 2480
= 8020
OR
Address of S[i][j]=BaseAddress +
W[(i-L1) + (j - L2) *M]
Address of S[15] [10] =
BaseAddress+ 4[(15–0)+(10 - 0)*40]
7200= Base Address + 4 [415]
Base Address = 7200 - 4 * 415

= 7200 - 1660
= 5540
Address of S[20][15]
= 5540 + 4 [(20 - 0) + (15 - 0) x 40]
= 5540 + 4 x 620
= 5540 + 2480
= 8020
OR
Address of Sri] [j] along the column =
Base Address + W [(i - L1) + (j - L2) * M]
XII Computer Chapter – 9 112
Address of S[15)[10] =
BaseAddress + 4[(15 - 1)+(10-1) x 40]
7200= Base Address + 4 [374]
Base Address = 7200 - 4 x 374
= 7200 - 1496
= 5704
Address of 5[20)[15]
= 5704 + 4 [(20 - 1) + (15 - 1) x 40]
= 5704 + 4 x 579
= 5704 + 2316
= 8020

(b) An array T[50][20] is stored in the memory along thecolumn with each of the elements occupying 4 bytes. Find outthe base address and address of element T[30][15], if anelement T[25][10] is stored at the memory location 9800.

Ans)Loc(T[I][J]) = Base(T)+W(I+J*N)
Loc(T[25][10]) = Base(T)+4(25+10*50)
Base(T) = 9800-4*525
Base(T) = 9800-2100
Base(T) = 7700
Loc(T[30][15]) =
Base(T)+4(30+15*50)
Loc(T[30][15])
= 7700 + 4(30+15*50)
= 7700 + 4(30+750)
= 7700 + 4*780
= 7700 + 3120
= 10820
OR
Address of T[i][j]
=BaseAddress + W [(i - L1) + (j - L2) * M]
Address of T[25] [10] =
BaseAddress + 4[(25 - 0)+(10 - 0)*50]
9800 = Base Address + 4 [525]
Base Address = 9800 - 4 * 525
= 9800 - 21.00
= 7700
Address of T[30][15]
=7700 + 4 [(30 - 0) + (15 - 0) x 50]
= 7700 + 4 x 780
= 7700 + 3120
= 10820
OR
Address of T[i][j] along the column
=Base Address+ W[(i - L1)+(j - L2)* M]
Address of T[25][10]
=BaseAddress + 4[(25 - 1) +(10 -1)x50]
9800= Base Address + 4 [474]
Base Address
= 9800 - 4 x 474
= 9800 - 1896
= 7904
Address of T[30][15]
= 7904 + 4 [(30 - 1) + (15 - 1) x 50]
= 7904 + 4 x 729
= 7904 + 2916
= 10820

3.b) An array Arr[40][10] is store in the memory along thecolumn with each element occupying 4 bytes. Find out thebase address of the location Arr[3][6] if the locationArr[30][10] is stored at the address 9000.

Solution:
Address of Array[i][j] along the column =Base Address + W [(i -
L1) + (j - L2) * M]
where,
W = size of each location in bytes = 4
L1 = Lower Bound of rows = 0
L2 = Lower Bound of columns = 0
M = Number of rows per column = 40
Address of Array[30][10]
= Base Address + 4 * (30 + 10 * 40)
9000 = Base Address + 4 * 430
Base Address = 9000 - 4 x 430
= 9000 -1720
= 7280
Address of Array[3][6]
= 7280 + 4 * (3 + 6 * 40)
= 7280 + 4 * 243
= 7280 + 972
= 8252
OR
Address of Array[i][j] along the column = Base Address + W ((i -
L1) + (j - L2) * M)
where,
W = size of each location in bytes = 4
L1 = Lower Bound of rows = 1
L2 = Lower Bound of columns = 1
M = Number of rows per column = 40
Address of Array[30][10]
= Base Address + 4 * ((30 -1) +(10 -1) * 40)
9000 = Base Address + 4 * (29+9*40)
9000 = Base Address + 4 * (29+360)
9000 = Base Address + 4 * (389)
Base Address
= 9000 - 4 * 389
= 9000 -1556
= 7444
Address of Array[3][6]
= 7444 + 4 * ((3 -1) + (6 -1) * 40)

= 7444 + 4 * (2+5 * 40)
= 7444 + 4 * (2+200),
= 7444 + 4 * 202
= 7444 + 808
= 8252

OR

Address of Array[i][j] along the column =Address of Array[x][y]
+ W [(i-x) + (j - y) * M]
where,
W = size of each location in bytes = 4
M = Number of rows per column = 40
i , j = Index value of the unknown element
x , y = Index value of the known element
Address of Array[3][6]
= Address of Array[30][10]+ 4 [(3 - 30)
+(6 -10) * 40]
= 9000 + 4 [-27 -160]
= 9000 - 4 x 187 = 9000 -748 = 8252
XII Computer Chapter – 9 113

3.b)An array Array[20][15] is stored in the memory along thecolumn with each element occupying 8 bytes. Find out thebase address of the element Array[2][3] if the elementArray[4][5] is stored at the address 1000.

Solution:

Given Data: Aray [20][15] W=8 B=? R=20
C=15 Lr = 0 Lc = 0
Address of Array [2][3] =?
Address of Array[4][5] =1000.

Address of an element (I,J) in column major =B + W ((I-Lr) + R(J-Lc))

Therefore
1000=B+8*((4-0)+20(5-0))
1000=B+8*(4+20*5)
1000 =B+8*104
1000=B+832
B =1000-832
B =168
Therefore Address of
Array[2][3]=168+8*((2-0)+20(3-0))
=168+8*(2+20*3)
=168+8*62
=168+496
=664

3.b) An array MAT[30][10] is stored in the memory alongcolumn wise with each element occupying 8 bytes of thememory. Find out the Base address and the address ofelement MAT[20][5] , if the location MAT[3][7] is stored atthe address 1000.

Ans)

For Column wise allocation
Address of A[I][J]
= BA + W[(J –LBC) x M + (I - LBR)]
Where
BA = Base Address
W = Size of each element in bytes
= 8 bytes (given)
M = No. of rows in the 2D Array = 30
(given)
Address of MAT[5][7] given is 1000.

Assumption 1 : LBR=LBC=0

Therefore
1000 = BA + 8 (7 × 30 + 5)
= BA + 8 × 215
= BA + 1720
BA = 1000 – 1720 = -720
Therefore,Base Address = - 720
Thus, Address of MAT[20][5] = -720 + 8 (5 × 30 + 20)
= -720 + 8 × 170
= -720 + 1360
= 640

Assumption 2 : LBR=LBC=1

Therefore
1000 = BA + 8 [(7-1) × 30 +(5-1)]
= BA + 8[6 × 30 + 4]
= BA + 8 ×184
= BA + 1472
BA = 1000 – 1472
= -472
Therefore,Base Address = - 472
Thus, Address of MAT[20][5]
= -472 + 8 (4 × 30 + 19)
= -472 + 8 × 139
= -472 + 1112
= 640

b) An array P[20][30] is stored in the memory along thecolumn with each of the element occupying 4 bytes, find outthe Base Address of the array, if an element P[2][20] is storedat the memory location 5000.

Ans)

Given,
W=4
N=20
M=30
Loc(P[2][20]) =5000

Column Major Formula:

Loc(P[I][J]) = Base(P)+W*(N*J+I)
Loc(P[2][20]) = Base(P)+4*(20*20+2)
Base(P) = 5000 -4*(400+2)
=5000-1608
=3392

3.b) An array P[20][30] is stored in the memory along thecolumn with each of the element occupying 4 bytes, find outthe memory location for the element P[5][15], if an elementP[2][20] is stored at the memory location 5000.

Ans)

Given,
W=4
N=20
M=30
Loc(P[2][20])=5000

Column Major Formula:
Loc(P[I][J])=Base(P)+W*(N*J+I)
Loc(P[2][20])=Base(P)+4*(20*20+2)5000
=Base(P)+4*(400+2)
Base(P) = 5000- 1608
Base(P) = 3392
Loc(P[5][15]) =3392+4*(20*15+5)
=3392+4*(300+5)
=3392+1220
=4612

3.b) An array ARR[5][5] is stored in the memory with eachelement occupying 3 bytes of space. Assuming the baseaddress of ARR to be 1500, compute the address ofARR[2][4], when the array is stored :3.b)An array X[30][10] is stored in the memory with eachelement requiring 4 bytes storage. Find out the Base addressof X is 4500, find out memory locations of X[12][8] andX[2][14], if the content is stored along the row.

3.d) The array A[20][10] is stored in the memory with eachelement requiring one byte of storage if the base address of ais 0, determine the location of A[10][5] when the array A isstored by column major.

3.b) An array X[10][20] is stored in the memory with eachelement requiring 4 bytes of storage. If the Base address of thearray is 1000, calculate location of X[5][15] when the array Xis stored using column major order.

3.b) An array VAL[1…15][1…10] is stored in the memorywith each element requiring 4 bytes of storage. If the baseaddress of the array VAL is 1500, determine the location ofVAL[12][9] when the array VAL is stored (i) Row wise (ii)Column wise.

Solution: Given Data:
VAL[1…15][1…10]
Word Length (W) = 4 Bytes
Base Address of VAL(B) = 1500
VAL[12][9] = ?
C = Total No of Columns
R = Total No of Rows
Lr = Least Row=1
Lc = Least Column=1

(i) Row Major:

Address of an element (I,J) in row major = B + W (C (I-Lr) +
(J – Lc))

VAL [12][9] = 1500 + 4 (10 * (12-1) + (9-1))
= 1500 + 4 (10 * 11+8)
= 1500 + 4 (118)
= 1500 + 472

(i) Column Major:

Address of an element (I,J) in column major
= B + W ((I-Lr) + R(J – Lc))

VAL [12][9] = 1500 + 4 ((12-1) +15 * (9-1))
= 1500 + 4 (11 + 15 * 8)
= 1500 + 4 (11+ 120)
= 1500 + 4 * 131
= 1500 + 524
= 2024.

3.b) An array A[10][20] is stored in the memory with eachelement requiring 4 bytes of storage. If the base address of thearray in the memory is 400, determine the location ofA[8][13] when the array VAL is stored (i) Row major (ii)Column major.

MODEL 4: Sorts & Search

3. (a) Write a function SORTPOINTS( ) in C++ to sort anarray of structure Game in descending order of Points using Bubble Sort.

Sample content of the array (before sorting)

PNo	PName	Points
103	Ritika Kapur	3001
104	John Philip	2819
101	Razia Abbas	3451
105	Tarun Kumar	2971

Sample content of the array (after sorting)

PNo	PName	Points
101	Razia Abbas	3451
103	Ri tika Kapur	3001
105	Tarun Kumar	2971
104	John Philip	2819

Note: Assume the following definition of structure Game

```
struct Game
{ long PNo; //Player Number
char PName [20] ;
long Points;
};
```

Ans)

```
void SORTPOINTS(Game G[], int N)
{ Game Temp;
for (int I = 0; I<N-l; I++)
for (int J = 0; J<N-I-l; J++)
if(G[J].Points < G[J+l].Points)
{
Temp = G[J];
G[J] = G[J+l];
G[J+l] = Temp;
}
}
```

(a) Write a function SORTSCORE( ) in C++ to sort an arrayof structure Examinee in descending order of Score using Bubble Sort.

Sample Content of the array (before sorting)

RollNo	Name	Score
1001	Ravyank Kapur	300
1005	Farida Khan	289
1002	Anika Jain	345
1003	George Peter	297

Sample Content of the array (after sorting)

RollNo	Name	Score
1002	Anika Jain	345
1001	Ravyank Kapur	300
1003	George Peter	297
1005	Farida Khan	289

Note: Assume the following definition of structure Examinee

```
struct Examinee
{ long RollNo;
char Name[20] ;
float Score;
} ;
```

Ans)

```
void SORTSOORE (Examinee E[ ], int N)
{ Examinee Temp;
for (int I = 0; I<N-l; I++)
for (int J = 0; J<N-I-l; J++)
if(E[J].Score < E[J+l].Score)
{
Temp = E[J];
E[J] = E[J+l];
E[J+l] = Temp;
}
}
```

3.a)Assume a array E containing elements of structureEmployee is required to be arranged in descending order ofSalary. Write a C++ function to arrange same with the help ofbubble sort, the array and its size is required to be passed asparameters to the function. Definition of structrure Employeeis as follows:

```
Struct Employee
{ int Eno;
char name[25];
float Salary;};
```

Solution:

```
void bubble(Employee E[ ],int n)
{ int i,j;
Employee Etemp;
for(i=0;i<n;++i)
for(j=0;j<(n-1)-i ;j++)
if(E[j].salary<E[j+1].salary)
{ Etemp=E[j];
E[j]=E[j+1];
E[j+1]=temp;}
cout<<"The details of the employee
in ascending order of salary ";
for(i=0;i<n;i++)
cout<<E[i].Eno<<'\t'<<E[i].name
<<'\t<<E[i].Salary<<endl;}
```

3.c) Considering the following key set: 42,29,74,11,65,58, use insertion sort to sort the data in ascending order and indicate the sequences of steps required. (2002)

Solution:

In this, Suppose an array A with n elements A[1],A[2],...A[N] is

in memory. The insertion sort algorithm scans A from A[1] to
A[N], insertion each element A[K] into its proper position in the
previously sorted subarray A[1],A[2],...,A[K-1].
This sorting algorithm is frequently used when n is small.
The array contains 6 elements as follows:
42,29,74,11,65,58

Pass	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
K=1	-32768	**42**	29	74	11	65	58
K=2	-32768	42	**29**	74	11	65	58
K=3	-32768	29	42	**74**	11	65	58
K=4	-32768	29	42	74	**11**	65	58
K=5	-32768	11	29	42	74	**65**	58
K=6	-32768	11	29	42	65	74	**58**
Sorted	**-32768**	**11**	**29**	**42**	**58**	**65**	**74**

3.a) Given two arrays of integers X and Y of sizes m and nrespectively. Write a function named MERGE() which willthird array named Z, such that the following sequence is followed.
(i) All odd numbers of X from left to right are copied into
from left to right
(ii) All even numbers of X from left to right are copied intoZ from right to left.
(iii) All odd numbers of Y from left to right are copied into Zfrom left to right.
(iv) All even numbers of Y from left to right are copied intoZ from right to left.
X, Y and Z are passed as arguments to MERGE( ).
Eg. X is {3, 2, 1, 7, 6, 3} and {9, 3, 5, 6, 2, 8, 10}
The resultant array Z is{3, 1, 7, 3, 9, 3, 5, 10, 8, 2, 6, 6, 2}
Ans)

```
void MERGE(int X[ ], int m,int Y[ ], int n,int Z[ ])
{ int mn,i,,left=0,right=mn-1;
mn=m+n;
for(i=0;i<m;i++)
if (X[i]%2= = 1)
Z[left++]=X[i];//For copying odd numbers of
                //X into Z from left to right else
Z[right- -]=X[i];//For copying even number of
                 //X into Z from right to left
for(i=0;i<n;i++)
if (X[i]%2= = 1)
Z[left++]=Y[i];
//For copying odd numbers of
//Y into Z from left to right
else
Z[right- -]=Y[i];
//For copying even number of
// X into Z from right to left
}
```

3.a) Suppose A, B, C are arrays of integers of size M, N and M+Nrespectively. The numbers in array A appear in ascending orderwhile numbers in array in descending order. Write user definedfunction in C++ to produce third array C by merging array A by Bin ascending order. Use A, B and C as arguments in the function.

```
void Merge(int A[ ],int M,int B[ ], int N,int C[ ])
{
int a,b,c;
for(a=0,b=N-1,c=0;a<M&&b>=0;)
{
if(A[a]<=B[b])
C[c++]=A[a++];
else
C[c++]=B[b--];
}
if(a<M)
{
while(a<M)
C[c++]=A[a++];
}
else
{
while(b>=0)
C[c++]=B[b--];
}
}
```

3.a) Suppose a 1D array AR containing integers is arranged inascending order. Write a user defined function in C++ tosearch for one integer from AR with the help of binary searchmethod, to show presence of the number in the array. Thefunction should have three parameters: (1) an array AR (2) the number to be searched and (3) the number of elements Nin the array.

```
void BinSearch(int AR[ ], int Sno, int N)
{ int l=0,u=N-1,m,flag=0;
while(l<=u)
{ m=(l+u)/2;
if (Sno= = AR[m])
{ flag=1;
break;
}
else if(Sno<AR[m])
u=m-1;
else
l=m+1;
}
if( flag = = 0)
cout<<"\nThe Search Element
" <<Sno<<" is not available";
else
cout<<"\nThe Search Element
" <<Sno<<" is available";
}
```

3.a) Suppose an array P containing float is arranged in
ascending order. Write a user defined function in C++ to

search for one float from p with the help of binary search
method. The function should return an integer 0 to show
absence of the number in the array. The function should have
the parameters as (1) an array P (2) the number DATA to be
searched (3) number of elements N. (1998)

```
int BinSearch(float P[ ], float DATA, int N)
{ int l=0,u=N-1,m;
while(l<=u)
{
m=(l+u)/2;
if (DATA= = P[m])
return 1;
else if(DATA<P[m])
u=m-1;
else
l=m+1;
}
return 0;
}
```

3.a) Write a function in C++ to merge the contents of two
sorted arrays A & B into third array C. Assuming array A
and B are sorted in ascending order and the resultant array C
is also required to be in ascending order. 3 (MP109-10)
Ans)

```
void AddNSave(int A[ ],int B[ ],int C[ ],
int N,int M, int &K)
{ int I=0,J=0;
K=0;
while (I<N && J<M)
if (A[I]<B[J])
C[K++]=A[I++];
else if (A[I]>B[J])
C[K++]=B[J++];
else
{
C[K++]=A[I++];
J++;
}
for (;I<N;I++)
C[K++]=A[I];
for (;J<M;J++)
C[K++]=B[J];
}
```

Q3. Write a function in C++ to merge the contents of two
sorted arrays A & B into third array C. Assuming array A is
sorted in ascending order, B is sorted in descending order, the
resultant array is required to be in ascending order. (MP108-
09) 4
Answer:

```
void AddNSave(int A[],int B[],int C[],int N,int M, int &K)
{ int I=0,J=M-1;
K=0;
while (I<N && J>=0)
{ if (A[I]<B[J])
C[K++]=A[I++];
else if (A[I]>B[J])
C[K++]=B[J--];
else
{
C[K++]=A[I++];
J--;
}
}
for (int T=I;T<N;T++)
C[K++]=A[T];
for (T=J;T>=0;T--)

C[K++]=B[T];
}
```

3.a) Write a function in C++, which accepts an integer array and its size as parameters and rearranges the array in reverse.

Example:If an array of nine elements initially contains the elements as 4, 2, 5, 1, 6, 7, 8, 12, 10
Then the function should rearrange the array as
10,12, 8, 7, 6, 1, 5, 2, 4

Solution:

```
void receive(int A[ ], int size)
{  int temp;
    for(i=0,j=size-1;i<size/2;i++,j--)
    {  temp=A[i];
       A[i]=A[j];
       A[j]=temp;
     }
 }//end of receive function.
```

3.b) An array Arr[40][10] is store in the memory along the column with each element occupying 4 bytes. Find out the base address of the location Arr[3][6] if the location Arr[30][10] is stored at the address 9000.

3.d) Write a function in C++ to print the product of each column of a two dimensional array passed as the arguments of the function.

1	2	4
3	5	6
4	3	2
2	1	5

Example : If the two dimensional array contains
Then the output should appear as:
Product of Column 1 = 24
Product of Column 2 = 30

```
Product of Column 3 =240
void receive(int A[ ][ ],int r,int c)
{  int i,j,B[c];
   for(i=0;i<c;i++)
  B[i]=1;
   for(i=0;i<r;i++)
     for(j=0;j<c;j++)
          B[j]=B[j]*A[i][j];
   for(i=0;i<c;i++)
         cout<<"\nProduct of Column "<<i+1<<"
        = "<<B[i];
 }
```

OUTSIDE DELHI 2008

3.a) Write a function in C++, which accepts an integer array and its size as arguments and swap the elements of every even location with its following odd location.

Example : If an array of nine elements initially contains the elements as 2,4,1,6,5,7,9,23,10

then the function should rearrange the array as

4,2,6,1,7,5,23,9,10

```
void SwapArray(int A[ ], int N)
{   int i,j,temp;
   /*  cout<<"\nThe elements before doing the
         desired alterations…";
      for(i=0;i<N;i++)
            cout<<A[i]<<'\t';  */
   for(i=0;i<N-1;i+=2)
   {   temp=A[i];
       A[i]=A[i+1];
       A[i+1]=temp;
   }
   /*  cout<<"\nThe elements after completed the
         desired alterations…";
      for(i=0;i<N;i++)       cout<<A[i]<<'\t';  */
}
```

3.b) An array Arr[50][10] is store in the memory along the row with each element occupying 2 bytes. Find out the Base address of the location Arr[20][50], if the location Arr[10][25] is stored at the address 10000.

Solution: Children, Try this answer as an assignment.

```
        else
        a[i]=a[i]*2;
        cout<<a[i]<<',';
    }
  }
```

3.b) An array Array[20][15] is stored in the memory along the **column** with each element occupying 8 bytes. Find out the base address of the element Array[2][3] if the element Array[4][5] is stored at the address 1000.

Solution:

Given Data: Aray [20][15] W=8 B=? R=20 C=15 $L_{r=0}$ $L_{c=0}$

Address of Array [2][3] =?

Address of Array[4][5] =1000.

Address of an element (I,J) in column major =B + W ((I-L_r) + R(J-L_c))

Therefore 1000=B+8*((4-0)+20(5-0))

1000=B+8*(4+20*5)

1000 =B+8*104

1000=B+832

B =1000-832

3.d) Write a function in C++ to print the product of each row of a two dimensional array passed as the arguments of the function

Example: if the two imensional array contains

Then the output should appear as:

```
Product of Row 1 = 8000
Product of Row 2 = 6000
Product of Row 3 =3600
Product of Row 4 = 2400
void receive(int A[ ][ ],int r,int c)
{  int i,j,B[r];
   for(i=0;i<r;i++)
  B[i]=1;
   for(i=0;i<r;i++)
     for(j=0;j<c;j++)
          B[i]=B[i]*A[i][j];
   for(i=0;i<r;i++)
  cout<<"\nProduct of Row "<<i+1<<" = "<<B[i];
 }
```

DELHI 2007

3.a) Write function in C++ which accepts an integer array and size as arguments and replaces elements having odd values with thrice its value and elements having even values with twice its value.

Example : if an array of five elements initially contains elements as 3, 4, 5, 16, 9

The the function should rearrange the content of the array as 9, 8, 75, 32, 27

Solution:

```
void manipulate (int a[ ],int size)
{ for (i=0;i<size;i++)
  {   if (a[i]%2= =1)
      a[i]=a[i]*3;
```

B =168

Therefore Address of Array[2][3]=168+8*((2-0)+20(3-0))

=168+8*(2+20*3)

=168+8*62

=168+496

=664

3.d) Write a function in C++ which accepts a 2D array of integers and its size as arguments and displays the elements which lie on diagonals. [Assuming the 2D Array to be a square matrix with odd dimension i.e., 3x3, 5x5 ,7x7 etc…]

20	40	10
40	50	30
60	30	20
40	20	30

Example : if the array content is

5 4 **3**

6 **7** 8

1 2 **9**

Out put through the function should be :

Diagonal One : 5 7 9

Diagonal Two : 3 7 1

Solution:

```
void accept(int a[ ][ ],int size)
{  cout<<"Diagonal One:";
   for (int i=0;i<size;i++)
        for(int j=0;j<size;j++)
            if (i= = j)
              cout<<a[i][j]<<'\t';
   cout<<"\n Diagonal Two:";
   for (i=0;i<size;i++)
        for(j=0;j<size;j++)
            if((i+j)= =(size-1))
              cout<<a[i][j]<<'\t';
}
```

OUTSIDE DELHI 2007

3.a) Write a function in C++ which accepts an integer array and its size as arguments and replaces elements having even values with its half and elements having odd values with twice its value .

Example : If an array of five elements initially contains the elements as 3, 4, 5, 16, 9

then the function should rearrange content of the array as

6, 2, 10, 8, 18

Solution:

```
void accept(int a[ ],int size)
{  for (int i=0;i<size;i++)
   {   if (a[i]%2= =0)
          a[i]=a[i]/2;
       else
              a[i]=a[i]*2;
       cout<<a[i]<<',';
    }
}
```

3.b) An array Arr[15][20] is stored in the memory along the **row** with each element occupying 4 bytes. Find out the Base address of the location Arr[3][2], if the location Arr[5][2] is stored at the address 1500.

Solution: Given Data: Arr[15][20] W=4 B=?
R=15 C=20 $L_r = 0$ $L_c = 0$

Address of Arr[3][2] = ?

Address of Arr[5][2] = 1500.

Address of an element (I,J) in row major = B+W(C(I-L_r)+(J-L_c))

Therefore, 1500 = B+4(20(5-0)+(2-0))
1500 = B+4(20*5+2)
1500 = B+4*102
1500 = B+408
B =1500-408
B=1092

Address of Arr[3][2] =1092+4(20*3+2)
=1092+4(62)
=1092+248 =1340.

3.d) Write a function in C++ which accepts a 2D array of integers and its size as arguments and displays the elements of middle row and the elements of middle column. [Assuming the 2D Array to be a square matrix with odd dimension i.e., 3x3, 5x5, 7x7 etc…]

Example : If the array content is

3 5 4
7 6 9
2 1 8

Output through the function should be :

Middle Row : 7 6 9
Middle Column : 5 6 1

Solution:

```
void accept(int a[ ][ ],int size)
{  cout<<"Middle Row:";
   for (int i=0;i<size;i++)
        for(int j=0;j<size;j++)
            if (i= = size/2)
              cout<<a[i][j]<<'\t';
   cout<<"\n Middle Column:";
   for (i=0;i<size;i++)
        for(j=0;j<size;j++)
            if(j= =size/2)
              cout<<a[i][j]<<'\t';
}
```

DELHI 2006

3.a) Write function in C++ which accepts an integer array and size as arguments and assign values into a 2D array of integers in the following format :

If the array is 1, 2, 3, 4, 5, 6

The resultant 2D array is given below

1 2 3 4 5 6
1 2 3 4 5 0
1 2 3 4 0 0
1 2 3 0 0 0
1 2 0 0 0 0
1 0 0 0 0 0

If the array is 1, 2, 3

The resultant 2D array is given :

1 2 3
1 2 0
1 0 0

Solution:

```
void input (int a[ ],int size)
{  int b[size] [size];
   for (int i=0;i.<size;i++)
   {
     for (int j=0;j<size;j++)
      {
        if(( i+j)>=size)
              b[i][j]=0;
        else
              b[i][j]=a[j];
       cout<<b[i][j]<<'\t';
      }
     cout<<endl;
   }
}
```

3.b) An array MAT[30][10] is stored in the memory along column wise with each element occupying 8 bytes of the memory. Find out the Base address and the address of element MAT[20][5] , if the location MAT[3][7] is stored at the address 1000.

Solution: Children, Try this answer as an assignment.

OUTSIDE DELHI 2006

3.a) Write function in C++ which accepts an integer array and size as arguments and assign values into a 2D array of integers in the following format :

If the array is 1, 2, 3, 4, 5, 6

The resultant 2D array is given below :

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

If the array is 1, 2, 3

The resultant 2D array is given :

```
1 0 0
1 2 0
1 2 3
```

Solution:

```
void input (int a[ ],int size)
{  int b[size] [size];
  for (int i=0;i.<size;i++)
  {  for (int j=0;j<size;j++)
     { if(( i<j)
            b[i][j]=0;
       else
            b[i][j]=a[j];
        cout<<b[i][j]<<'\t';
      }
     cout<<endl;
   }
}
```

3.b) An array MAT[20][10] is stored in the memory along the row with each element occupying 4 bytes of the memory. Find out the Base address and the address of element MAT[10][5] , if the location MAT[3][7] is stored at the address 1000.

Solution: Children, Try this answer as an assignment.

DELHI 2005

3.a) Write a function in C++ which accepts an integer array and its size as arguments and exchanges the values of first half side elements with the second half side elements of the array.

Example :

If an array of 8 elements initial content as 2, 4, 1, 6, 7, 9, 23, 10

The function should rearrange array as 7, 9, 23, 10, 2, 4, 1, 6

Solution:

```
void change(int a[ ],int size)
{
  int i,j,temp;
  for(i=0,j=size/2;j<size;i++,j++)
  {  temp=a[i];
     a[i]=a[j];
     a[j]=temp;
  }
}
```

3.b) An array Arr[15][35] is stored in the memory along the row with each of its element occupying 4 bytes . Find out the Base address and the address of element Arr[2][5] , if the location Arr[5][10] is stored at the address 4000.

Solution: Children, Try this answer as an assignment.

3.d) Write a function in C++ to print sum of all values which either are divisible by 2 or divisible by 3 present in a 2D array passed as the argument of the function.

Solution:

```
void Sum(int A[ ][ ],int R,int C)
{   int i,j,S=0;
    for(i=0;i<R;i++)
        for(j=0;j<C;j++)
            if(A[i][j]%2= = 0 ||A[i][j]%3= = 0)
                S=S+A[i][j];
    cout<<"\nThe Sum of all the values which are divisible by 2 or 3 in the array = "<<S;
}
```

OUTSIDE DEHI 2005

3.a) Write a function in C++ which accepts an integer array and its size as arguments and exchanges the values of first half side elements with the second half side elements of the array.

Example :

If an array of 8 elements initial content as 8, 10, 1, 3, 17, 90, 13, 60

The function should rearrange array as 17, 90, 13, 60, 8, 10, 1, 3

Solution: Refer Delhi 2005 Q.3a.

3.b) An array Arr[35][15] is stored in the memory along the row with each of its element occupying 4 bytes . Find out the Base address and the address of element Arr[20][5] , if the location Arr[2][2] is stored at the address 3000.

Solution: Children, Try this answer as an assignment.

3.d) Write a function in C++ to print sum of all values which either are divisible by 3 or divisible by 5 present in a 2D array passed as the argument of the function.

Ans:-

```
void Sum(int A[ ][ ],int R,int C)
{  int S=0,i,j;
  for(i=0;i<R;i++)
     for(j=0;j<C;j++)
        if((a[i][j]%3= =0)||(a[i][j]%5= =0))
           S=S+A[i][j];
  cout<<" nThe Sum of all the values which are divisible by 3 or 5 in the array = "<<S;
}
```

DELHI 2004

3.a) Define the function **SwapArray(int[ ], int),**that would expect a 1D integer array NUMBERS and its size N. the function should rearrange the array in such a way that the values of that locations of the array are exchanged. (Assume the size of the array to be even).

Example :

If the array initially contains {2, 5, 9, 14, 17, 8, 19, 16}

Then after rearrangement the array should contain {5, 2, 14, 9, 8, 17, 16, 19}

Solution:

```
void SwapArray(int NUMBERS[ ], int N)
{   int i,j,temp;
    /*  cout<<"\nThe elements before doing the
          desired alterations…";
        for(i=0;i<N;i++)
              cout<<NUMBERS[i]<<'\t';  */
    for(i=0;i<N-1;i+=2)
    {   temp=NUMBERS[i];
        NUMBERS[i]=NUMBERS[i+1];
        NUMBERS[i+1]=temp;
    }
   /*  cout<<"\nThe elements after completed the
          desired alterations…";
        for(i=0;i<N;i++)
              cout<<NUMBERS[i]<<'\t';  */
}
```

3.b) An array ARR[5][5] is stored in the memory with each element occupying 3 bytes of space. Assuming the base address of ARR to be 1500, compute the address of ARR[2][4], when the array is stored :

Solution: Children, Try this answer as an assignment.

3.c) Write a function in C++ to find the sum of diagonal elements from a 2D array of type float. Use the array and its size as parameters with float as its return type.

Solution:

```
float diasum(float A[ ][ ],int R,int C)
{  int i,j;
   float Dsum=0.0;
   for(i=0;i<R;i++)
       for(j=0;j<C;j++)
           if((i= = j)| | (i+j)= =(size-1))
               Dsum=Dsum+A[i][j];
  return Dsum;
}
```

DELHI 2003

3.a) Assume a array E containing elements of structure Employee is required to be arranged in descending order of Salary. Write a C++ function to arrange same with the help of bubble sort, the array and its size is required to be passed as parameters to the function. Definition of structrure Employee is as follows:

```
Struct Employee
{
    int Eno;
    char name[25];
    float Salary;
};
```

Solution:

```
void bubble(Employee E[ ],int n)
{ int i,j;
  Employee Etemp;
  for(i=0;i<n;++i)
     for(j=0;j<(n-1)-i ;j++)
          if(E[j].salary<E[j+1].salary)
           {          Etemp=E[j];
                      E[j]=E[j+1];
                      E[j+1]=temp;
           }
    cout<<"The details of the employee in ascending
          order of salary ";
    for(i=0;i<n;i++)

          cout<<E[i].Eno<<'\t'<<E[i].name<<'\t<<E[
          i].Salary<<endl;
   }
```

3.b) An array X[30][10] is stored in the memory with each element requiring 4 bytes storage. Find out the Base address of X is 4500, find out memory locations of X[12][8] and X[2][14], if the content is stored along the row.

Solution: Children, Try this answer as an assignment.

3.c) Write a user-defined function in C++ to display those elements of 2D array T[4][4] which are divisible by 100. Assume the content of the array is already present and the function prototype is as follows: void showhundred(int T[4][4]);

```
void showhundred(int T[4][4])
{  int i,j;
   cout<<"\nThe elements in the array which are
         divisible by 100 …..";
   for(i=0;i<4;i++)
       for(j=0;j<4;j++)
          if(T[i][j]%100= =0)
               cout<<T[i][j]<<'\t';
}
```

DELHI 2002

3.a) Define array and pointer.

Solution: An array refer to a named list of a finite number n of similar data elements. Each of the data elements can be referenced respectively by a set of consecutive numbers. Arrays can be one dimensional, two dimensional or multi dimensional.

An array can be declared as : Syntax: data_type Array_name[size];

Eg: int A[10]; //Then location of the array are A[0], A[1],…….A[9]. int B[5][4]; //This array can holds 5 X 4 = 20 elements.

3.d) The array A[20][10] is stored in the memory with each element requiring one byte of storage if the base address of a is 0, determine the location of A[10][5] when the array A is stored by column major.

Solution: Children, Try this answer as an assignment.

3.c) Considering the following key set: 42,29,74,11,65,58, use insertion sort to sort the data in ascending order and indicate the sequences of steps required.

Solution:

In this, Suppose an array A with n elements A[1],A[2],…A[N] is in memory. The insertion sort algorithm scans A from A[1] to A[N], insertion each element A[K] into its proper position in the previously sorted subarray A[1],A[2],…,A[K-1].

This sorting algorithm is frequently used when n is small.

The array contains 6 elements as follows:
42,29,74,11,65,58

Pass	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
K=1	-32768	**42**	29	74	11	65	58
K=2	-32768	42	**29**	74	11	65	58
K=3	-32768	29	42	**74**	11	65	58
K=4	-32768	29	42	74	**11**	65	58
K=5	-32768	11	29	42	74	**65**	58
K=6	-32768	11	29	42	65	74	**58**
Sorted	**-32768**	**11**	**29**	**42**	**58**	**65**	**74**

DELHI 2001

3.a) Given two arrays of integers X and Y of sizes m and n respectively. Write a function named MERGE() which will third array named Z, such that the following sequence is followed.

(i) All odd numbers of X from left to right are copied into Z from left to right.
(ii) All even numbers of X from left to right are copied into Z from right to left.
(iii) All odd numbers of Y from left to right are copied into Z from left to right.
(iv) All even numbers of Y from left to right are copied into Z from right to left.

X, Y and Z are passed as arguments to MERGE().
Eg. X is {3, 2, 1, 7, 6, 3} and {9, 3, 5, 6, 2, 8, 10}
the resultant array Z is {3, 1, 7, 3, 9, 3, 5, 10, 8, 2, 6, 6, 2}

```
void MERGE(int X[ ], int m,int Y[ ],int n,int Z[ ])
{   int mn,i,,left=0,right=mn-1;
    mn=m+n;
    for(i=0;i<m;i++)
        if (X[i]%2= = 1)
          Z[left++]=X[i]; //For copying odd
        numbers of X into Z from left to right
        else
          Z[right- -]=X[i]; //For copying even
        number of X into Z from right to left
   for(i=0;i<n;i++)
        if (X[i]%2= = 1)
          Z[left++]=Y[i];   //For copying odd
        numbers of Y into Z from left to right
        else
          Z[right- -]=Y[i]; //For copying even
        number of X into Z from right to left
}
```

3.b) An array X[10][20] is stored in the memory with each element requiring 4 bytes of storage. If the Base address of the array is 1000, calculate location of X[5][15] when the array X is stored using column major order.

NOTE: X[10][20] means valid row indices are 0 and 9 and valid column indices are 0 and 19

Solution: Children, Try this answer as an assignment.

3.c) Write a user-defined function named Lower_half() which takes 2D array A, with size N rows and N columns as argument and prints the lower half of the array.

```
Eg.    2  3  1  5  0                          2
       7  1  5  3  1                          7  1
Input  2  5  7  8  1      the output will be   2  5
       7
       0  1  5  0  1                          0  1
       5  0
       3  4  9  1  5                          3  4
       9  1  5
```

Solution:

```
void Lower_half( int A[ ][ ],int N)
{   int i,j;
   for(i=0;i<N;i++)
       for(j=0;j<N;j++)
       {   if(i<j)
               cout<<A[i][j]<<'\t';
           cout<<endl;
       }
}
```

DELHI 2000

3.a) Suppose A, B, C are arrays of integers of size M, N and M+N respectively. The numbers in array A appear in ascending order while numbers in array in descending order. Write user defined function in C++ to produce third array C by merging array A by B in ascending order. Use A, B and C as arguments in the function.

```
void Merge(int A[ ],int M,int B[ ],int N,int C[ ])
{   int a,b,c;
   for(a=0,b=N-1,c=0;a<M&&b>=0;)
   {       if(A[a]<=B[b])
                C[c++]=A[a++];
           else
                C[c++]=B[b--];
    }
    if(a<M)
    {       while(a<M)
                C[c++]=A[a++];
     }
     else
     {      while(b>=0)
                C[c++]=B[b--];
      }
}
```

3.b) An array VAL[1…15][1…10] is stored in the memory with each element requiring 4 bytes of storage. If the base address of the array VAL is 1500, determine the location of VAL[12][9] when the array VAL is stored (i) Row wise (ii) Column wise.

Solution:
Given Data:
VAL[1…15][1…10]
Word Length (W) = 4 Bytes
Base Address of VAL(B) = 1500
VAL[12][9] = ?

C = Total No of Columns R = Total No of Rows

L_r = Least Row=1 L_c = Least Column=1

(i) Row Major:

Address of an element (I,J) in row major = B + W (C (I-L_r) + (J – L_c))

VAL [12][9] = 1500 + 4 (10 * (12-1) + (9-1))

= 1500 + 4 (10 * 11+8)

= 1500 + 4 (118)

= 1500 + 472

= 1972.

(i) Column Major:

Address of an element (I,J) in column major = B + W ((I-L_r) + R(J – L_c))

VAL [12][9] = 1500 + 4 ((12-1) +15 * (9-1))

= 1500 + 4 (11 + 15 * 8)

= 1500 + 4 (11+ 120)

= 1500 + 4 * 131

= 1500 + 524

= 2024.

3.c) Write a user-defined function in C++ to find and display the sum of diagonal elements from a 2D array MATRIX[6][6] containing integers.

```
void displaysum( )
{  int i,j,D1=0,D2=0,MATRIX[6][6];
  cout<<"\nEnter any 36 values….";
  for(i=0;i<6;i++)
      for(j=0;j<6;j++)
       {  cin>>MATRIX[i][j];
          if(i= = j)
              D1=D1+MATRIX[i][j];
          else if ((i+j)= =(size-1))
              D2=D2+MATRIX[i][j];
       }
    cout<<"\nThe sum of the elements of the Main
Diagonal = "<<D1;
    cout<<"\nThe sum of the elements of the Other
Diagonal = "<<D2;
}
```

DELHI 1999

3.a) Suppose a 1D array AR containing integers is arranged in ascending order. Write a user defined function in C++ to search for one integer from AR with the help of binary search method, to show presence of the number in the array. The function should have three parameters: (1) an array AR (2) the number to be searched and (3) the number of elements N in the array.

```
void BinSearch(int AR[ ], int Sno, int N)
{   int  l=0,u=N-1,m,flag=0;
    while(l<=u)
    {   m=(l+u)/2;
        if (Sno= = AR[m])
         {    flag=1;
              break;
         }
        else if(Sno<AR[m])
              u=m-1;
        else
              l=m+1;
    }
    if( flag = = 0)
          cout<<"\nThe Search Element
"<<Sno<<" is not available";
    else
          cout<<"\nThe Search Element
"<<Sno<<" is available";
 }
```

3.b) An array A[10][20] is stored in the memory with each element requiring 4 bytes of storage. If the base address of the array in the memory is 400, determine the location of A[8][13] when the array VAL is stored (i) Row major (ii) Column major.

Solution: Children, Try this answer.

3.c) Write a user-defined function in C++ to find and display the multiplication of row elements of two dimensional array A[4][6] containing integers.

```
void rowmul( )
{   int A[4][6],i,j,rowmul;
    cout<<"\nEnter any 24 values…";
    for(i=0;i<4;i++)
        for(j=0;j<6;j++)
            cin>>A[i][j];
  for(i=0;i<4;i++)
   {    rowmul=1;
        for(j=0;j<6;j++)
              rowmul=rowmul*A[i][j];
        cout<<"\nThe multiplication of "<<i+1<<"
row = "<<rowmul;
    }
  }
```

DELHI 1998

3.a) Suppose an array P containing float is arranged in ascending order. Write a user defined function in C++ to search for one float from p with the help of binary search method. The function should return an integer 0 to show absence of the number in the array. The function should have the parameters as (1) an array P (2) the number DATA to be searched (3) number of elements N.

```
int BinSearch(float P[ ], float DATA, int N)
{   int  l=0,u=N-1,m;
    while(l<=u)
    {   m=(l+u)/2;
        if (DATA= = P[m])
              return 1;
        else if(DATA<P[m])
              u=m-1;
        else
              l=m+1;
    }
    return 0;
 }
```

3.b) An array T[15][10] is stored in the memory with each element requiring 2 bytes of storage. If the base address of

T is 2000, determine the location of T[7][8] when the array VAL is stored (i) Row major (ii) Column major.

Solution: Children, Try this as an assignment.

3.c) Write a user-defined function in C++ to find and display the sum of diagonal elements from a 2D array R[7][7] containing integers.

```
void displaysum( )
{  int i,j,D1=0,D2=0,R[7][7];
   cout<<"\nEnter any 49 values….";
   for(i=0;i<7;i++)
       for(j=0;j<7;j++)
       {  cin>>R[i][j];
          if(i= = j)
              D1=D1+R[i][j];
          else if ((i+j)= =(size-1))
              D2=D2+R[i][j];
       }
   cout<<"\nThe sum of the elements of the Main Diagonal = "<<D1;
   cout<<"\nThe sum of the elements of the Other Diagonal = "<<D2;
}
```

9.ARRAYS (8 Marks)

MODEL 1: Function to Receive an array and ChangeElements. (2 or 3 Marks)

Write a user-defined function swap_row(int ARR[ ][3],int R,intC) in C++ to swap the first row values with the last row values:**(2017MP) 2**

For example if the content of the array is:

10 20 30
40 50 60
70 80 90

Then after function call, the content of the array should be:

70 80 90
40 50 60
10 20 30

Ans)

```
void swap_row(int ARR[][3],int R,int C)
{
for(int i=0,j=0;j<C;j++)
{
int temp=ARR[i][j];
ARR[i][j]=ARR[R-1][j];
ARR[R-1][j]=temp;
}
}
```

Write the definition of a function grace_score (int score [], intsize) in C++, which should check all the elements of the arrayand give an increase of 5 to those scores which are less than40.

Example: if an array of seven integers is as follows:

45, 35, 85, 80, 33, 27, 90

After executing the function, the array content should bechanged as follows:

45, 40, 85, 80, 38, 32, 90 **(2016)3**

Ans)

```
void grace_score(int score[],int size)
{
for(int i=0;i<size;i++)
{
if(score[i]<40)
score[i]=score[i]+5;
cout<<score[i]<<" ";
}
}
```

Write the definition of a function FixSalary(float Salary[], int N) in C++, which should modify each element of the array Salary having N elements, as per the following rules: (2016) 2

A)

Existing Salary Values	Required Modification in Value
If less than 100000	Add 35% in the existing value
If >=100000 and <20000	Add 30% in the existing value
If >=200000	Add 20% in the existing value

```
{
for (int i=0;i<N;i++)
if(Salary[i]<100000)
Salary[i]+= 0.35 *Salary[i];
else if (Salary[i]>=100000 && Salary[i]<20000)
Salary[i]+= 0.3 * Salary[i];
else if(Salary[i]>=200000)
Salary[i]+= 0.20 * Salary[i];
}
```

Write the definition of a function Change(int P[], int N) in C++, which should change all the multiples of 10 in the array to 10and rest of the elements as 1. For example, if an array of 10 integers is as follows: (2015)2

P[0]	P[1]	P[2]	P[3]	P[4]	P[5]	P[6]	P[7]	P[8]	P[9]
100	43	20	56	32	91	80	40	45	21

After executing the function, the array content should be changed as follows:

P[0]	P[1]	P[2]	P[3]	P[4]	P[5]	P[6]	P[7]	P[8]	P[9]
10	1	10	1	1	1	10	10	1	1

A)

```
void Change(int P[ ],int N)
{
for (int i=0;i<N;i++)
if(P[i]%10==0)
P[i]=10;
else
P[i]=1;
}
```

Write a code for function EvenOdd(int T[ ], int C) in C++, to add 1 in all the odd values and 2 in all the even values of thearray T.

T[0]	T[1]	T[2]	T[3]	T[4]
35	12	16	69	26

The modified content will be:

T[0]	T[1]	T[2]	T[3]	T[4]
36	14	18	70	28

(Answer)

```
void EvenOdd(int T[ ],int C)
{
int I;
for(i=0;i<C;i++)
{
if(T[i]%2==0)
T[i]=T[i]+2;
else
T[i]=t[i]+1;
}
cout<<"Modified content will be: ";
for(i=0;i<C;i++)
cout<<T[i];
}
```

Write code for a function void ChangOver(int P[ ],int N) inC++, which repositions all the elements of the array byshifting each of them to the next position and by shifting thelast element to the first position. (2013) 3

For example: if the content of array is

0	1	2	3	4
12	15	17	13	21

The changed content will be

0	1	2	3	4
21	12	15	17	13

(Ans)

```
void Change(int P[ ], int N)
{
int temp;
int temp1;
for(int i=0;i<(N-1);i++)
{
temp=P[size-1];
P[N-1]=P[i];
P[i]=temp;
}
}
```

3. (a) Write a function SWAP2BEST (int ARR[], int Size) inC++ to modify the content of the array in such a way that theelements, which are multiples of 10 swap with the valuepresent in the very next position in the array. (2012) 3For example:

If the content of array ARR is
90, 56, 45, 20, 34, 54
The content of array ARR should become
56, 90, 45, 34, 20, 54

Ans

```
void SWAP2BEST(int ARR[], int Size)
{
int t;
for(int i=0;i<Size-1;i++)
{ if (ARR[i] %10=0)
{ t=ARR[i];
ARR[i]=ARR[i+1];
ARR[i+1]=t;
i++; //Ignore if not. written
}
}
}
```

Write a Get2From1() function in C++ to transfer the contentfrom one array ALL[] to two different arrays Odd[] andEven[]. The Odd[] array should contain the values from oddpositions (1,3,5,...) of ALL[] and Even [] array should containthe values from even positions (0, 2, 4,.....) of ALL [].(2011 OD) 3

Example

If the ALL[] array contains
12, 34, 56, 67, 89, 90
The Odd[] array should contain
34, 67, 90
And the Even [] array should contain
12,56,89

Ans

```
void Get2From1 (int All [],int Even [], int Odd [], int Size)
{int J=0,K=0;
for (int I=0 ;I<Size; 1++)
{ if (I%2==0)
{
Even [J]=All[I] ;
J++;
}
else
{
Odd[K]=All[I];
K++;
}
}
}
```

3.(a) Write a function CHANGEO in C++, which accepts anarray of integer and its size as parameters and divide all thosearray elements by 7 which are divisible by 7 and multiplyother-array elements by 3. (2010D)
Sample Input Data of the arrayContent of the array after Calling CHANGE( ) function

Ans)

Sample Input Data of the array

A[0]	A[1]	A[2]	A[3]	A[4]
21	12	35	42	18

Content of the array after Calling CHANGE( ) function

A[0]	A[1]	A[2]	A[3]	A[4]
3	36	5	6	54

```
void CHANGE (int A[ ], int N)
{
for(int I = 0; I<N; I++)
{
```

```
if (A[I]%7 = = 0)
A [I] = A [I] /7;
else
A[I] = A[I] * 3;
}
}
```

3. (a) Write a function REASSIGNO in C++, which accepts anarray of integers and its size as parameters and divide allthose array elements by 5 which are divisible by 5 andmultiply other array elements by 2.
(2010OD)Sample Input Data of the array
Content of the array after calling REASSIGNO function

Sample Input Data of the array

A[0]	A[1]	A[2]	A[3]	A[4]
20	12	15	60	32

Content of the array after calling REASSIGNO function

A[0]	A[1]	A[2]	A[3]	A[4]
4	24	3	12	64

Ans)

```
void REASSIGN (int Arr[ ], int Size)
{
for (int i=0;i<Size;i++)
if (Arr[i]%5==0)
Arr[i]/=5;
else
Arr[i]*=2;
}
```

OR

```
void REASSIGN(int Arr[ ],int Size)
{
for (int i=0;i<Size;i++)
Arr[i]%5 ? Arr[i]/=5 :
Arr[i] *= 2;
}
```

(d) Define a function SWAPCOL ( ) in C++ to swap (interchange) the first column elements with the last columnelements, for a two dimensional integer array passed as theargument of the function. (2009 D)

Example: If the two dimensional array contains

2	1	4	9
1	3	7	7
5	8	6	3
7	2	1	2

After swapping of the content of 1st column and last column, it should be:

9	1	4	2
7	3	7	1
3	8	6	5
2	2	1	7

Ans)

```
void SWAPCOL(int A[][100], int M, int N)
{ int Temp, I;
for(I=O; I<M; I++)
{
Temp = A [I][0] ;
A[I][0] = A[I][N-l];
A[I][N-l] = Temp;
}
}
```

OR

```
void SWAPCOL(int A[4][4])
{
int Temp, I;
for(I=O; I<4; I++)
{
Temp = A[I][0];
A[I][0] = A[I][3];
A[I][3] = Temp;
}
}
```

(d) Define a function SWAPARR( ) in C++ to swap (interchange) the first row elements with the last row elements, for a two dimensional integer array passed as theargument of the function.

Example: If the two dimensional array contains

5	6	3	2
1	2	4	9
2	5	8	1
9	7	5	8

After swapping of the content of first row and last row, it should be as follows:

9	7	5	8
1	2	4	9
2	5	8	1
5	6	3	2

(2009 OD)

Ans)

```
void SWAPARR(int A[][100], int M, int N)
{
int Temp, Ji
for (J=0; J<N; J++)
{
Temp = A[0)[J] ;
A[0][J] = A[M-1][J];
A[M-l][J] = Temp;
}
}
```

OR

```
void SWAPARR(int A[4][4])
{
int Temp, J;
for (J=0; J<4; J++)
{
Temp = A[0][J];
A[0][J] = A[3][J];
A[3][J] = Temp;
}
```

```
}
```

3.a) Write a function in C++, which accepts an integer array and its size as parameters and rearranges the array in reverse. (2008D)

Example:
If an array of nine elements initially contains the elements as 4, 2, 5, 1, 6, 7, 8, 12, 10
Then the function should rearrange the array as 10,12, 8, 7, 6, 1, 5, 2, 4

Solution:

```
void receive(int A[ ], int size)
{ int temp;
for(i=0,j=size-1;i<size/2;i++,j--)
{ temp=A[i];
A[i]=A[j];
A[j]=temp;
}
} //end of receive function.
```

3.a)Write a function in C++, which accepts an integer arrayand its size as arguments and swap the elements of every evenlocation with its following odd location. (2008OD)Example :
If an array of nine elements initially contains the elements as2,4,1,6,5,7,9,23,10
then the function should rearrange the array as 4,2,6,1,7,5,23,9,10

```
void SwapArray(int A[ ], int N)
{
int i,j,temp;
/* cout<<"\nThe elements before doing the desired alterations…";
for(i=0;i<N;i++)
cout<<A[i]<<'\t'; */
for(i=0;i<N-1;i+=2)
{ temp=A[i];
A[i]=A[i+1];
A[i+1]=temp;
}
```

XII Computer Chapter – 9 100

```
/* cout<<"\nThe elements after completed the desired alterations…";
for(i=0;i<N;i++)
cout<<A[i]<<'\t'; */
}
```

3.a)Write function in C++ which accepts an integer array andsize as arguments and replaces elements having odd valueswith thrice its value and elements having even values withtwice its value.(2007D)

Example : if an array of five elements initially contains elements as 3, 4, 5, 16, 9
The the function should rearrange the content of the array as 9, 8, 75, 32, 27

Solution:

```
void manipulate (int a[ ],int size)
{
for (i=0;i<size;i++)
{
if (a[i]%2= =1)
a[i]=a[i]*3;
else
a[i]=a[i]*2;
cout<<a[i]<<',';
}
}
```

3.a)Write a function in C++ which accepts an integer arrayand its size as arguments and replaces elements having evenvalues with its half and elements having odd values with twiceits value .(2007OD)

Example : If an array of five elements initially contains the elements as 3, 4, 5, 16, 9
then the function should rearrange content of the array as 6, 2, 10, 8, 18

Solution:

```
void accept(int a[ ],int size)
{
for (int i=0;i<size;i++)
{ if (a[i]%2= =0)
a[i]=a[i]/2;
else
a[i]=a[i]*2;
cout<<a[i]<<',';
}
}
```

3.a)Write function in C++ which accepts an integer array andsize as arguments and assign values into a 2D array of integersin the following format : (2006D)

If the array is 1, 2, 3, 4, 5, 6
The resultant 2D array is given below
1 2 3 4 5 6
1 2 3 4 5 0
1 2 3 4 0 0
1 2 3 0 0 0
1 2 0 0 0 0
1 0 0 0 0 0

If the array is 1, 2, 3
The resultant 2D array is given :
1 2 3
1 2 0
1 0 0

Solution:

```
void input (int a[ ],int size)
{ int b[size] [size];
for (int i=0;i.<size;i++)
{
for (int j=0;j<size;j++)
{
if(( i+j)>=size)
```

```
b[i][j]=0;
else
b[i][j]=a[j];
cout<<b[i][j]<<'\t';
}
cout<<endl;
}
}
```

3.a)Write function in C++ which accepts an integer array andsize as arguments and assign values into a 2D array of integersin the following format : (2006OD)

If the array is 1, 2, 3, 4, 5, 6

The resultant 2D array is given below :

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

If the array is 1, 2, 3

The resultant 2D array is given:

```
1 0 0
1 2 0
1 2 3
```

Solution:

```
void input (int a[ ],int size)
{
int b[size] [size];
for (int i=0;i.<size;i++)
{
for (int j=0;j<size;j++)
{
if(( i<j)
b[i][j]=0;
else
b[i][j]=a[j];
cout<<b[i][j]<<'\t';
}
cout<<endl;
}
}
```

OR

```
const int R = 100, C = 100;
void Arrayconvert(int A1D[ ], int N)
{ int A2D[R][C]={0};
for(int I = 0; I<N; I++)
for (int J = 0; J <=I; J++)
A2D[I][J] = A1D[J];
}
```

XII Computer Chapter – 9 101

3.a)Write a function in C++ which accepts an integer arrayand its size as arguments and exchanges the values of first halfside elements with the second half side elements of the array.(2005D)

Example :

If an array of 8 elements initial content as
2, 4, 1, 6, 7, 9, 23, 10
The function should rearrange array as
7, 9, 23, 10, 2, 4, 1, 6

Solution:

```
void change(int a[ ],int size)
{ int i,j,temp;
for(i=0,j=size/2;j<size;i++,j++)
{ temp=a[i];
a[i]=a[j];
a[j]=temp;
}
}
```

OR

```
void Exchange (int A [ ], int N)
{ for (int I=0;I<N/2;I++)
{ int Temp=A[I];
A[I]=A[N/2+I];
A[N/2+I]=Temp;
}
}
```

OR

```
void Exchange(int A[], int N)
{ int M=(N%2==0)?N:N+l;
for (int I=0; I<M/2; I++)
{ int Temp=A[I];
A[I]=A[M/2+I];
A[M/2+I]-Temp;
}
}
```

3.a)Write a function in C++ which accepts an integer arrayand its size as arguments and exchanges the values of first halfside elements with the second half side elements of the array.(2005OD)

Example :

If an array of 8 elements initial content as
8, 10, 1, 3, 17, 90, 13, 60
The function should rearrange array as
17, 90, 13, 60, 8, 10, 1, 3

Ans)

```
void Exchange(int A[],int N)
{ for (int I=0;I<N/2;I++)
{ int Temp=A[I];
A[I]=A[N/2+I];
A[N/2+I]=Temp;
}
}
```

OR

```
void Exchange(int A[],int N)
{ for (int I=0,J=N/2;I<N/2;I++,J++)
{ int Temp=A[J];
for (int K=J;K>I;K--)
A[K]=A[K-1];
A[I]=Temp;
}
}
```

OR

```
void Exchange(int A[],int N)
{ int M=(N%2=0)?N:N+l;
for (int I=0;I<M/2;I++)
{
int Temp=A[I];
A[I]=A[M/2+I];
A[M/2+I]=Temp;
}
}
```

3.a) Define the function SwapArray(int[ ], int),that would expect 1D integer array NUMBERS and its size N. the function should rearrange the array in such a way that the values of that locations of the array are exchanged. (Assume the size of the array to be even). (2004)

Example :

If the array initially contains {2, 5, 9, 14, 17, 8, 19, 16}
Then after rearrangement the array should contain {5, 2, 14, 9, 8, 17, 16, 19}

Solution:

```
void SwapArray(int NUMBERS[ ], int N)
{
int i,j,temp;
/* cout<<"\nThe elements before doing the desired alterations…";
for(i=0;i<N;i++)
cout<<NUMBERS[i]<<'\t'; */
for(i=0;i<N-1;i+=2)
{
temp=NUMBERS[i];
NUMBERS[i]=NUMBERS[i+1];
NUMBERS[i+1]=temp;
}
/* cout<<"\nThe elements after completed the desired alterations…";
for(i=0;i<N;i++)
cout<<NUMBERS[i]<<'\t'; */
}
```

3.c) Write a user-defined function in C++ to find and displaythe sum of diagonal elements from a 2D array MATRIX[6][6]containing integers.

```
void displaysum( )
{
int i,j,D1=0,D2=0,MATRIX[6][6];
cout<<"\nEnter any 36 values….";
for(i=0;i<6;i++)
for(j=0;j<6;j++)
{ cin>>MATRIX[i][j];
if(i= = j)
D1=D1+MATRIX[i][j];
else if ((i+j)= =(size-1))
D2=D2+MATRIX[i][j];
}
cout<<"\nThe sum of the elements of
the Main Diagonal = "<<D1;
cout<<"\nThe sum of the elements of
the Other Diagonal = "<<D2;
}
```

XII Computer Chapter – 9 102

3.a) Write a function in C++ to combine the contents of twoequi-sized arrays A and B by adding their correspondingelements as the formula A[i]+B[i]; where value i varies from 0to N-1 and transfer the resultant content in the third samesized array C. (MP209-10) 3

Ans)

```
void AddNSave(int A[ ],int B[ ],int C[ ],int N)
{ for (int i=0;i<N;i++)
C[i]=A[i]+B[i];
}
```

3.a)Write a function in C++ to combine the contents of twoequi-sized arrays A and B by computing their correspondingelements with the formula2*A[i]+3*B[i]; where value i variesfrom 0 to N-1 and transfer the resultant content in the thirdsame sized array. (MP208-09) 4

Ans)

```
void AddNSave(int A[ ],int B[ ],int C[ ],int N)
{
for (int i=0;i<N;i++)
C[i]=2*A[i]+3*B[i];
}
```