DIVYA CHRISTOPHER Module 3 Notes

(i) Arrays -

An array is a collection of data items belonging to the same data type.

Array indexing always starts from 0.

Array size is specified within a single subscript (index) denoted by one square bracket [].

Each array element is referred by specifying the array name followed by one or more subscripts in square brackets [][].

In a one-dimensional (1-D) array, a row of values are stored in contiguous locations in memory.

Syntax of 1-D array -

storage-class datatype array-name [size];

size is a positive integer expression enclosed in square brackets.

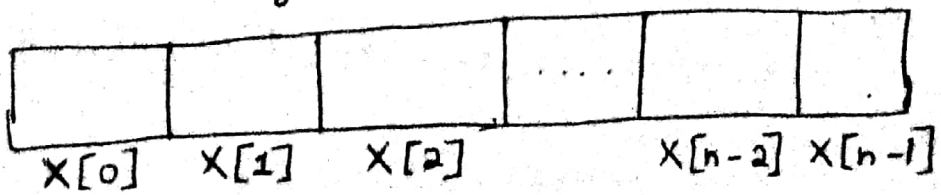
Eg:- int a[10];

Storage-class specification is optional, it refers to the storage class of an array.

The above declaration indicates an integer array containing 10 elements, ranging from 0 to 9.

But, an array could not store different types of values together.

In an n-element array, array elements are X[0], X[1], ..., X[n-1]



Examples of 1-D array -

Eg① - int digits [10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

digits [0] = 1

digits [1] = 2

digits [2] = 3

digits [3] = 4

digits [4] = 5

digits [5] = 6

digits [6] = 7

digits [7] = 8

digits [8] = 9

digits [9] = 10

Each array individual element is stored in contiguous locations in memory.

Eg ② :- int digits[5] = {3, 3, 3};

digits[0] = 3

digits[1] = 3

digits[2] = 3

digits[3] = 0

digits[4] = 0

Eg ③ :- int digits[] = {1, 2, 3, 4};

digits[0] = 1

digits[1] = 2

digits[2] = 3

digits[3] = 4

Eg:- char color[4] = {'R', 'E', 'D', '\0'};

A string is represented as a 1-D char type array. Each character within the string is stored within one array element.

color[0] = 'R'

color[1] = 'E' color[3] = '\0';

color[2] = 'D'

Write a C program to convert a sentence from Lower case to Upper case-

```
#include<stdio.h>
#include<ctype.h>
#define size 80
int main()
{
char letter[size];
 int count;
for (count=0; count<size; ++count)
   letter[count] = getchar();
for(count = 0; count<size; ++count)
  putchar (toupper (letter[count]));
}
```

where the first and second subscripts denote the number of rows and columns.

In a multi-dimensional array, a separate pair of square brackets is required for each subscript. A 2-D array will require 2 pairs of square brackets, a 3-D array will require 3 pairs of square brackets and so on.

A 2-D array can be a table of m rows and n columns.

Eg:- int values[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
//values is a 2-D array having 3 rows and 4 columns.

Here is a variation of the 2D array definition.

```
int values[3][4] = {
                    {1,2,3,4};
                    {5,6,7,8};
                    {9,10,11,12};
                   };
```

Each array element takes up values as follows:-

values[0][0] = 1, values[0][1] = 2, values[0][2] = 3, values[0][3] = 4,

values[1][0] = 5, values[1][1] = 6, values[1][2] = 7, values[1][3] = 8.

values[2][0] = 9, values[2][1] = 10, values[2][2] = 11, values[2][3] = 12.

1-D

Eg of character array -

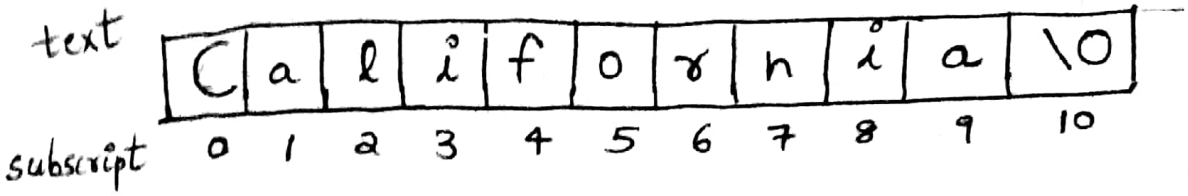
The 11-element character array representation is as shown Below:-

```
char text [] = "California";
```

\0 \zero } denotes end of string

or

```
char text[11] = "California";
```




```
char text[0] = 'C';
```

Note that an n-character string will require an (n+1)-element array, including the null character (\0) that is automatically placed at the end of the string.

- In a 2-D array or multidimensional array, a collection of data items are stored along both rows and columns.
- The 2D array size is specified using two subscripts, denoted by 2 square brackets [][], where the first and second subscripts denote the number of rows and columns.

Syntax of 2D array -

Storage class datatype array-variable-name [row size] [column size];

The default indexing of an array starts from 0.

Eg:- int A[3][3];

A is a 2-D array that can be thought of as a table of values containing 3 rows and 3 columns.

In the figure below, array indexing starts from 0.

A[0][0]	A[0][1]	A[0][2]
A[1][0]	A[1][1]	A[1][2]
A[2][0]	A[2][1]	A[2][2]

Consider if array indexing starts from 1, then the 2D matrix representation will become,

A[1][1]	A[1][2]	A[1][3]
A[2][1]	A[2][2]	A[2][3]
A[3][1]	A[3][2]	A[3][3]

The 2-D array elements will be stored as follows in a 2-D array.

```
int A[3][3]; // A is a 2-D array with 3 rows and 3 columns.
```

A[i][j] refers to an element in the 2D array.

10	11	12
13	14	15
16	17	18

A[0][1] = 11.

The above array declaration indicates an integer array containing 3 rows and 3 columns.

Qn) Write a C program to find the largest even or odd number in an array?

```
#include<stdio.h>
int main()
{int a[50], i, j, n, c, number, largeeven=0, largeodd=0;
 printf("Enter the array size");
  scanf("%od", &n);
  printf("Enter the array elements");
   for(i=0; i<n; i++)
   {
   scanf("%d", &a[i]);
   }
  largeeven = a[0];
  largeodd = a[0];
  for(j=0; j<n; j++)
  {
 if((a[j] & 1) == 0)
   {if(a[j] > largeeven)
    largeeven = a[j];
    printf("%d is even.", a[j]);
   }
   else
    {if(a[j] > largeodd)
     largeodd = a[j];
     printf("%d is odd.", a[j]);
}}
 printf("Largest even no:=%d",
          largeeven);
 printf("Largest odd no:=%od",
          largeodd);
 }
```

Output

```
Enter the array size 3
Enter the array elements
5 6 7
5 is odd. 6 is even.
7 is odd.
Largest even number=6
Largest odd number=7.
```

DIVYA CHRISTOPHER

Array Operations

1) Searching an Array

o Linear Search

In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found, then that particular item is returned, otherwise search continues till the end of the data collection.

Linear Search eg

Element	10	14	19	26	27	31	33
Index	0	1	2	3	4	5	6

There are a total of 7 elements in the array.
The search element 31 is found in position 5.

Default array indexing starts from 0 and ends at n-1.

Linear Search Algorithm-

Start

Step 1 :

Step 2: integer pos, i, count=0, a[10], n, s

Step 3: Read the limit, n

Step 4: Input the array elements

for(i ← 0; i < n; i++)

Read a[i]

Step 5: Input the number to be searched.

Read s.

Step 6:

```
for(i←0; i<n; i++)
{
if (a[i] == s)
{
  count ++;
   pos= i+1;
}
}
```

Step 7:

```
if (count == 0)
print "Number not found"
else
print "Number", s, "is found
at position", pos
```

Step 8: Stop

Pseudocode for Linear Search -

```
Read the limit, n
Input the array elements as a list
Input the number to be searched.
for each array element in the list
    if array element == search value
        return the array element found
            with location in an array
    end if
end for
End.
```

Linear Search Program –

```
#include <stdio.h>
int main()
{
int pos, i, count=0, a[20], n, s;
printf("Enter the limit");
scanf("%d", &n);
printf("Enter the array elements");
for (i=0; i<n; i++)
    scanf("%d", &a[i]);
printf("Enter the element you
    coant to search");
scanf("%d", &s);
for (i=0; i<n; i++)
  {
    if (a[i]==s)
      {
        count = count+1;
        pos= i+1;
      }
  }
```

```
if(count == 0)
    printf("Number not found");

else
    printf ("Number %d is found
    at position %d", s, pos);
}
```

Output

```
Enter the limit 7
Enter the array elements
10 14 19 26 27 31 33
Enter the element you
want to search 31
Number 31 is found at position 6.
```

2) Sorting an Array

(a) Bubble Sort

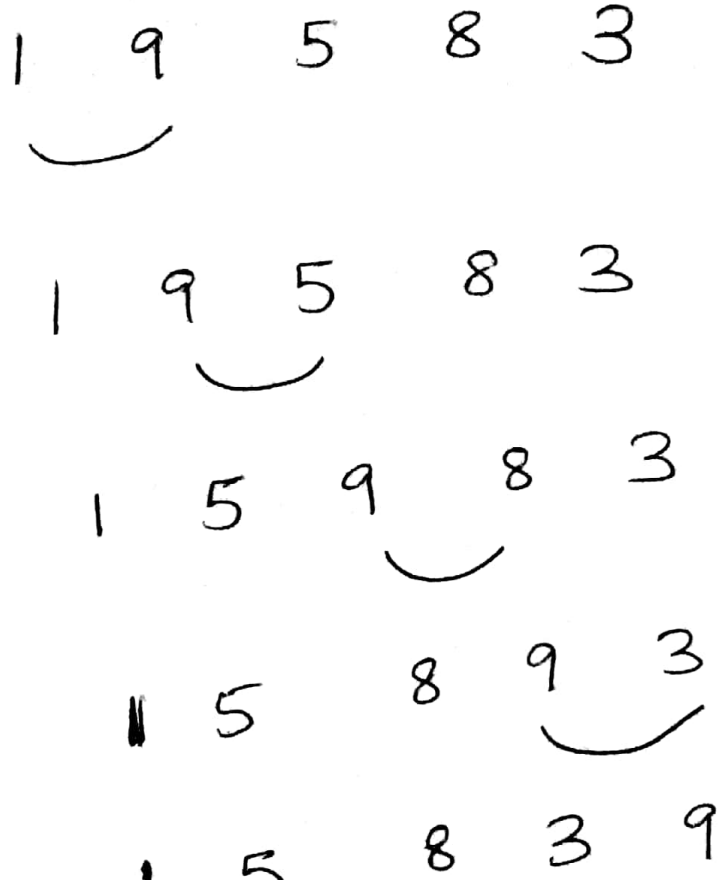
It is a sorting algorithm which is a comparison-based algorithm, in which each pair of adjacent elements are compared and the elements are swapped, if they are not in order.

Eg:-

The array elements are

1 9 5 8 3

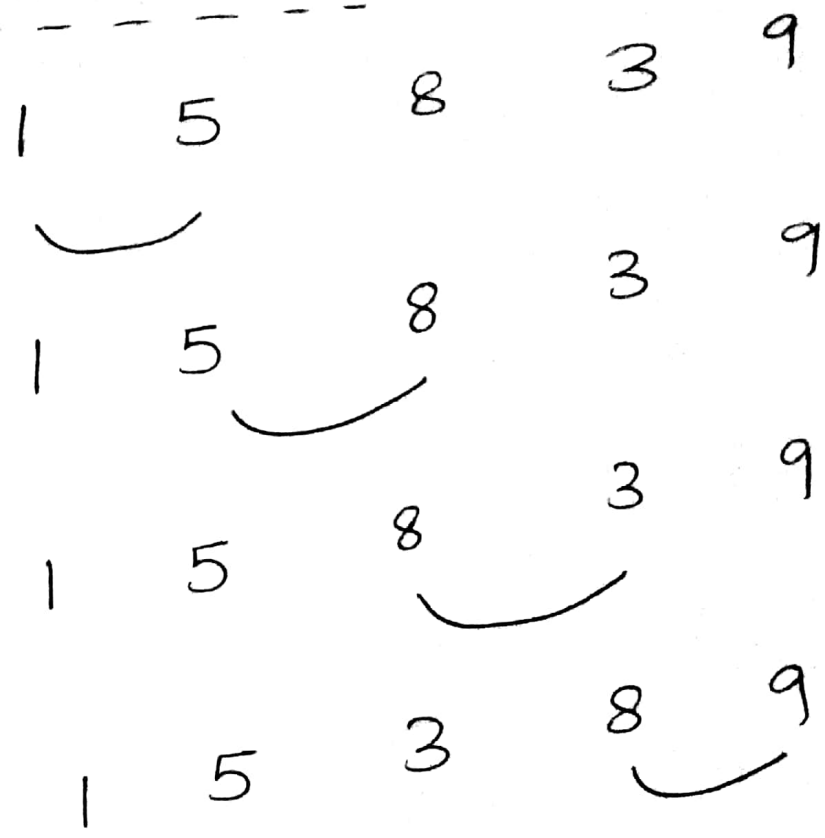
First Pass - Algorithm compares the first 2 elements, and swaps if first number is greater than second number.



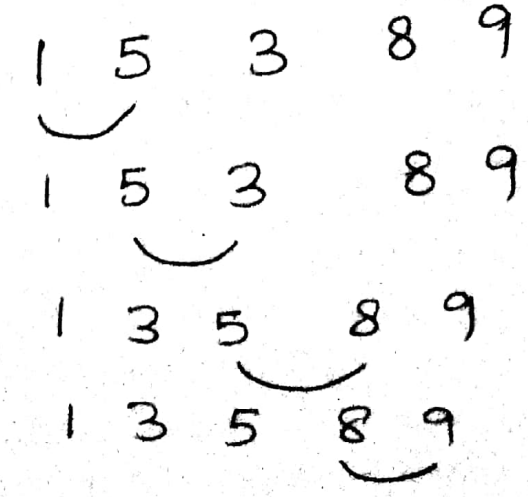
From the beginning of the list, Algorithm again compares the first 2 elements, and swaps if first number is greater than second number.

The sorting process continues until the entire list is sorted.

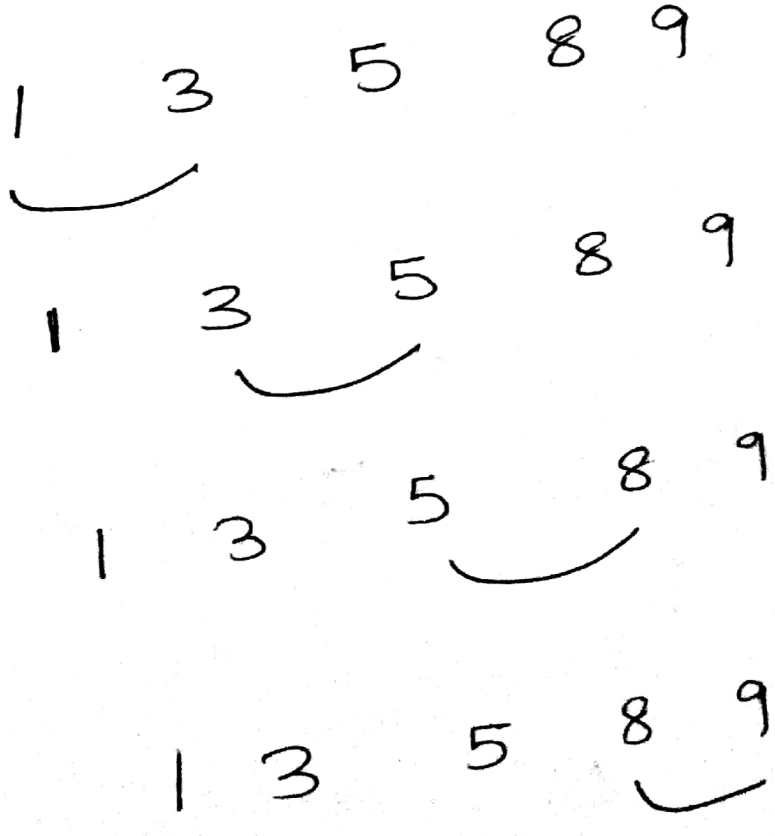
Second Pass -



Third Pass - From the beginning of the list, algorithm again compares the first two elements, and swaps if first number is greater than second number in the list



Fourth Pass - From the beginning of the list, algorithm again computes one whole pass, without any swap, to ensure that the list is sorted.



Finally, the list is sorted.

Algorithm - Bubble Sort

sorting in Ascending order

Step 1. Start

Step 2. integer i, n, ..., a[10], temp

Step 3. Read the limit, n.

Step 4. Input the array elements

```
for (i ← 0 ; i < n; i++)
    Read a[i]
```

Step 5.

```
for (i ← 0 ; i < n ; i++)
{
  for (j ← i+1 ; j < n; j++)
  {
    if (a[i] > a[j])
    {
      temp = a[i];
      a[i] = a[j];
      a[j] = temp;
    }
  }
}
```

Step 6. Print the array elements

```
for (i ← 0 ; i < n; i++)
    Print a[i]
```

Step 7. Stop

Pseudocode for Bubble Sort

Read the limit, n

Input the array elements as a list.

for all elements of the list

```
for (i = 0; i < n; i++)
    for (j = i+1; j < n; j++)
        if a[i] is greater than a[j]
            Swap a[i] and a[j]
        end if
    end for
end for
Print the array elements as a list
end.
```

(*Note - In Descending order sorted list, we compare whether a[i] is lesser than a[j] to perform swap between array elements.)

Bubble Sort - Program

```
#include <stdio.h>
int main()
{
int a[10], i, j, temp, n;
printf("Enter the limit");
scanf("%d", &n);
printf("Enter the array elements");
for(i=0; i<n; i++)
scanf("%d", &a[i]);
for(i=0; i<n; i++)
{
for(j=i+1; j<n; j++)
{
if(a[i]>a[j])
{
temp = a[i];
a[i] = a[j];
a[j] = temp;
}
}
}
```

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

Output

Enter the limit 5

Enter the array elements

45

21

89

78

99

The Sorted list is

21 45 78 89 99

STRING PROCESSING - Characters are stored and processed in C as strings.

In-built string handling functions -
(strlen, strcpy, strcmp, strcat)

String is an array of characters.

All string operations (functions) for string length computation, string copy, string comparison, and string concatenation are included in string.h header file

Eg:- char str1[] = {'A', 'B', 'C', 'D', '\0'};

char str1[] = "ABCD";

\0 could be automatically inserted at the end in this declaration.

① String length - strlen()

This function returns the length of the string including terminating character '\0'.

Syntax -

size_t strlen (const char *str);

Eg:-
```
#include <stdio.h>
#include <string.h>
int main()
{
char str1[20] = "LMCST";
printf ("Length of string : %d ", strlen(str1));
}
```

Output

Length of string : 5

② String compare function - strcmp() -

It compares 2 strings and returns an Integer value. If both the strings are equal, then this function coould return a 0, otherwise it may return a -ve or +ve value based on the comparison.

Syntax -

```
Int strcmp(const char *str1, const char *str2);
```

③ String concatenation function - strcat()

It concatenates two strings and returns the concatenated string.

Syntax -

```
char *strcat (char *str1, char *str2);
```

④ String copy function - strcpy()

char *strcpy (char *str1, char *str2)

It copies string str2 into str1, including the terminating character \0.

Eg:-
```
#include <stdio.h>
#include <string.h>
int main()
{
char s1[20] = "LMCST";
 char s2[20] = "LOURDES MATHA COLLEGE";
 if (strcmp (s1, s2) == 0)
   printf("Both strings are equal");
 else printf("Both strings are different");
 strcat (s1, s2);
 printf("\nString after concatenation: %s", s1);
 strcpy (s1, s2);
  printf ("Copied String is: %s", s1);
}
```

Output

Both strings are different.

String after concatenation: LMCST LOURDES MATHA COLLEGE

Copied String is LOURDES MATHA COLLEGE OLLEGE

Qn) write a C program to count the no: of words, vowels and white spaces in a given line of text?

```
#include <stdio.h>
int main()
{
char text[80], ch;
int vcount, wcount, whitecount, i;
printf("Enter the line of text");
gets(text);
puts(text);
i=0;
vcount = wcount = whitecount = 0;
while ((ch = text[i]) != '\0')
{
if (ch == 'a' || ch == 'e' || ch == 'i' ||
ch == 'o' || ch == 'u')
vcount ++;
else
if (ch == ' ')
{
wcount ++;
} whitecount ++;
```

```
i++;
}
printf("No: of vowels = %d", vcount);
printf("No: of words = %d", wcount);
printf("No: of whitespaces = %d", whitecount);
}
```

Qn) Write a program to check whether a given string is a palindrome or not?

```
#include <stdio.h>
int main()
{
char text[80];
int beg, back, l;
int flag;
printf("Enter the string");
scanf("%s", text);
l=0;
while (text[l] != '\0')
  l++;
  flag = 1;
  beg=0;
  back = l-1;
```

```
while ((beg <= back) && flag)
    {
    if (text [beg] == text [back])
     flag = 1 ;
     else
      flag = 0;
      beg++;
      back--;
    }
if(flag)
printf("The string is a palindrome");
else
printf("The string is not a
        palindrome");
}
```

Q) Write a C program to print the result of Matrix Addition?

```
#include <stdio.h>
int main()
{
int x[3][3], y[3][3], z[3][3], i, j, m, n;
  printf("Enter the no: of rows and columns");
  scanf("%d %d", &m, &n);
    printf("Enter matrix 1");
  for(i=0; i<m; i++)
  for(j=0; j<n; j++)
   scanf("%d", &x[i][j]);
   printf("Enter matrix 2");
    for(i=0;i<m; i++)
    for(j=0;j<n;j++)
     scanf("%d", &y[i][j]);
     printf("Matrix Addition Result\n");
     for(i=0; i<m; i++)
   { printf("\n");
   for(j=0; j<n; j++)
    {
    z[i][j] = x[i][j] + y[i][j];
    }
     printf("%d\t", z[i][j]);
    }
}
```

Qn) Write a C program to display the transpose of a matrix and to print matrix multiplication result?

```
#include <stdio.h>
int main()
{
int i,j, m,n, a[3][3], b[3][3], c[3][3];
printf ("Enter the order of matrix");
scanf ("%d %d", &m, &n);
printf ("Enter the array elements");
for (i=0; i<m; i++)
 {
 for (j=0; j<n; j++)
  {
  scanf ("%d", &a[i][j]);
 }
}
printf ("The array elements are");
for(i=0; i<m; i++)
 { printf ("\n");
 for(j=0; j<n; j++)
  {
    printf ("%d \t", a[i][j]);
 }
}
```

```
printf("Transpose of a matrix is");
for (i=0; i<m; i++)
 {
  for(j=0; j<n; j++)
   {
     printf("%d", a[j][i]);
   }
  }
 printf("Matrix multiplication result is");
  for(i=0; i<m; i++)
   {
  for(j=0; j<n; j++)
   { printf("\n");
   for(K=0; K<m; K++)
    {
   c[i][j] = c[i][j] + (a[i][K] * b[K][j]);
    }
     printf("%d\t", c[i][j]);
   }
 }
} // end of main
```

Qn) Write a C program to print the diggonal, lower and upper triangular elements in a matrix?

```
#include <stdio.h>
int main()
{
int a[10][10], s1, s2, s3, i, j, m, n;
printf("Enter the no: of rows
      and columns in a matrix");
scanf("%d %d", &m, &n);
printf("Enter the array elements");
for(i=0; i<m; i++)
{
for(j=0; j<n; j++)
{scanf("%d", &a[i][j]);
}
}
printf("Matrix is");
for(i=0; i<m; i++)
{ printf("\n");
for(j=0; j<n; j++)
 { printf("%d\t", a[i][j]);
}}
```

```
printf("The diagonal matrix is \n");
  for(i=0; i<m; i++)
   {  printf("\n");
  for(j=0; j<n; j++)
   {
   if(i==j)
     {
      printf("%d\t", a[i][j]);
      s1=s1+a[i][j];
  } } }
  printf("Sum of diagonal matrix: %d", s1);
printf("The upper triangular matrix is \n");
  for(i=0; i<m; i++)
   {
   printf("\n");
  for(j=0; j<n; j++)
    {if(i<j)
     {
      printf("%d\t", a[i][j]);
       s2=s2+a[i][j];
     }
  }
printf("Sum of upper triangular matrix: %d", s2);
```

```
printf ("Lower triangular matrix is\n");
 for (i=0 ; i< m; i++)
  {
   printf("\n");
  for (j= 0; j<n; j++)
     {
     if (i>j)
      {
      printf("%d\t", a[i][j]);
     s3 = s3 + a[i][j];
      }
  }
}
printf(" Sum of Lower Triangular
          Matrix : %d\n", s3);

 }
```

Qn) Write a C program to find the sum and average of n numbers in an array?

```
#include <stdio.h>
int main()
{  float avg;
 int sum=0, i, n, a[10];
 printf("Enter the number of elements in an array");
 scanf("%d", &n);
 printf("Enter the array elements");
 for(i=0; i<n; i++)
  scanf("%d", &a[i]);
 printf("The array elements are");
  for(i=0; i<n; i++)
   {
   printf("%d\t", a[i]);
    sum = sum+a[i];
   }
   avg=sum / float(n);
 printf("Sum = %d, Average = %d", sum, avg);
}
```

Qn) Write a C program to check whether a string is a Palindrome or not?

```
#include <stdio.h>
#include <string.h>
int main()
{
char a[25] , b[25];
int i,j, flag=0, n;
printf (" Enter the string");
 gets (a);
 n= strlen(a);
 j= 0;
for (i=n; i >0 ; i--)
 {
  b[j] = a [i];
  j++;
 }
for(i=0; i<n ; i++)
 {
 if (b[i] != a[i])
   {
    flag = 1;
    break;
   }
 }
if (flag == 1)
printf("Not a Palindrome
else
printf (" Palindrome");
}
```

Qn) Write a C program to find the Reverse of a line of text?

```
#include <stdio.h>
#include <string.h>
int main()
{
char s[25], a[25];
int i,j=0,n;
  printf("Enter the string");
  gets(s);
   n=strlen(s);
  for(i=n-1; i>=0; i--)
   {
    a[j]= s[i];
    printf("%c", a[j]);
    j++;
   }
}
```

J

Qn) Write a C program to implement string handling functions without built-in functions?

```
#include <stdio.h>
#include <string.h>
int main()
{
char str[50], str1[10], str2[10];
int i, len = 0, flag = 0, temp;
printf("Enter a string");
gets(str);
printf("Entered String is \n");
```

```
for(i=0; str[i] != '\0' ; i++)
  { printf("%c", text[i]);
  { len++;
  }
printf("Length of string %s is %d", str, len);
printf("STRING COMPARISON");
printf("Enters the first string");
  gets(str1);
printf("Enter the second string");
  gets(str2);
for(i=0 ; str1[i] != '\0' ; i++)
  {           if(str1[i] == str2[i])
                flag = 1;
              else
                flag = 0;
  }
  if(flag == 1)
     printf("Both strings are same\n");
  else  printf("Both strings are not same\n");
printf("STRING COPY");
for(i=0 ; str1[i] != '\0'; i++)
    str2[i] = str1[i];
    str2[i] = '\0';
printf("Copied string is: %s", str2);
```

```
printf ("Concatenated String is");
    for (i=0 ; str1 [i] != '\0' ; i++)
      len ++;
      temp = len;
    for (i=0 ; str2[i] != '\0' ; i++)
      {
       str1 [temp] = str2 [i];
       temp++;
      }
      str1 [temp] = '\0';
      printf ("The concatenated string is:");
      puts (str1);
}
```

Qn) Write a C program to find the smallest and largest element in an array?

```
#include <stdio.h>
#include <conio.h>
void main()
{
int i, n, a[10], small, large;
printf("Enter the limit");
scanf("%d", &n);
printf("Enter the array elements");
for (i=0; i<n; i++)
scanf("%d", &a[i]);
small = a[0];
large = a[0];
if (a[i] > large)
    large = a[i];
if (a[i] < small)
    small = a[i];
```

```
printf(" Smallest no in an array : %d", small);
printf ("Largest no: in an array: %d", large);
 getch();
}
```

DIVYA CHRISTOPHER

Linux Version

Qn) Write a C program to print Matrix addition result, Matrix Multiplication result and Matrix Transpose Result?

```
#include <stdio.h>
int main()
{
int x[3][3], y[3][3], z[3][3], i, j, k, m, n, p, q;
int a[3][3], b[3][3], c[3][3];

printf ("Enter the no: of rows and
                        columns");
scanf ("%d %d", &m, &n);
printf(" Enter matrix 1");
for (i=0; i<m; i++)
  {
for (j=0 ; j<n; j++)
  {
    scanf("%d", &x[i][j]);
  }
}
```

```
printf ("Enter matrix 2");
 for (i=0; i<m; i++)
  {
 for(j= 0; j<n; j++)
   {
   scanf("%d", &y[i][j]);
  }
 }
printf ("Matrix addition result");
 for (i=0; i<m; i++)
  {
   printf("\n");
   for(j=0 ;j<n ;j++)
    {
   z[i][j] = x[i][j] + y[i][j];
     }
   }
for (i=0 ;i<m ; i++)
 {
  printf("\n");
  for(j=0; j<n; j++)
    {
   printf("%d\t", z[i][j]);
 } }
```

```
printf(" MATRIX MULTIPLICATION PROCESSING");
printf (" Enter the no: of rows and columns
 of first matrix");
   scanf ("%d %d", &m, &n);
printf ("Enter the no: of rows and columns
 of second matrix");
   scanf ("%d %d", &p, &q);
   if(n == p)
    {
    printf (" Enter the elements of first
     matrix");
     for (i=0 ; i<m ; i++)
     {
     for (j=0; j<n; j++)
      {
       scanf("%d", &a[i][j]);
      }
     }
```

```
printf("Enter the elements of
second matrix");
for(i=0; i<p; i++)
{
for(j=0; j<q; j++)
{
scanf("%d", &b[i][j]);
}
}
printf("Matrix multiplication");
for(i=0; i<m; i++)
{
for(j=0; j<q; j++)
{
c[i][j]=0;
for(k=0; k<p; k++)
{
c[i][j]=c[i][j]+(a[i][k]*b[k][j]);
}
}
}
}//end-if
```

```
printf ("Matrix multiplication result");
  for (i=0; i<m; i++)
   { printf("\n");
   for (j=0; j<q; j++)
   {
     printf ("%d \t", c[i][j]);
   }
  }
  printf ("Transpose processing of a Matrix");
   for (i=0; i<m; i++)
    {
     for(j=0; j<n; j++)
      {
       z[i][j] = z[j][i];
      }
    }
   printf("Matrix Transpose Result");
    for(i=0; i<m; i++)
    {printf("\n");
     for (j=0; j<n; j++)
      {
        printf ("%d \t", z[i][j]);
      }
    }
```