ALAGAPPA UNIVERSITY

[Accredited with 'A+' Grade by NAAC (CGPA:3.64) in the Third Cycle
and Graded as Category–I University by MHRD-UGC]

(A State University Established by the Government of Tamil Nadu)

KARAIKUDI – 630 003

Directorate of Distance Education

B.C.A.

I - Semester

101 14

LAB: C AND DATA STRUCTURE

Reviewer	
Dr. K. Kuppusamy	Professor and Head (i/c), Department of Computational Logistics, Alagappa University, Karaikudi

Author

Dr. Kavita Saini, *Assistant Professor, School of Computer Science & Engineering, Galgotias University, Greater Noida.*

Vikas® is the registered trademark of Vikas® Publishing House Pvt. Ltd.

VIKAS® PUBLISHING HOUSE PVT. LTD.
E-28, Sector-8, Noida - 201301 (UP)
Phone: 0120-4078900 • Fax: 0120-4078999
Regd. Office: 7361, Ravindra Mansion, Ram Nagar, New Delhi 110 055
• Website: www.vikaspublishing.com • Email: helpline@vikaspublishing.com

Work Order No. AU/DDE/DE1-238/Preparation and Printing of Course Materials/2018 Dated 30.08.2018 Copies - 500

LAB: C AND DATA STRUCTURE

Syllabi

BLOCK 1: C PROGRAM FUNDAMENTALS

1. Simple C Programs
2. Using if and Switch Constructs Programs
3. Looping Statements Problems

BLOCK 2 : FUNCTIONS, ARRAYS, STRINGS, FILE AND POINTERS

4. Functions and Recursive Programs
5. Arrays, Strings and Matrices Programs
6. File Handling Programs
7. Pointers and Arrays Programs Programs

BLOCK 3: STRUCTURE, UNION AND FILES

8. Structure and Union: Programs using Structure and Union
9. Files: Programs Based on File Handling

BLOCK 4: LINEAR DATA STRUCTURE PROGRAMS

10. Stacks, Queues, Expression Evaluation Programs
11. Infix to Postfix Conversion
12. Linked List Programs: List, Merging Lists, Linked List, Single Linked List, Double Linked List, Header Linked List, Insertion and Deletion of Linked List, Traversing a Linked List.

BLOCK 5: NON LINEAR DATA STRUCTURE EXPERIMENTS

13. Tree Programs: Trees, Binary Trees, Types of Binary Trees, Binary Tree Representation
14. Traversing Binary Trees, Binary Search Tree, Insertion and Deletion Operations

INTRODUCTION

NOTES

In computing, C is a computer programming language developed between 1969 and 1973 by Dennis Ritchie at the Bell Telephone Laboratories to be used with the UNIX operating system. The data structure features are also supported by the C language. Data structures are the ability of a computer to fetch and store data at any place in its memory, specified by an address which is a bit string that can be itself stored in memory and manipulated by the program. The record and array data structures are based on computing the addresses of data items with arithmetic operations, while the linked data structures are based on storing addresses of data items within the structure itself. The implementation of a data structure usually requires writing a set of procedures that create and manipulate instances of that structure. With the advent of multi-core processors, many known data structures have concurrent versions that allow multiple computing threads to access the data structure simultaneously. A linked data structure is a data structure which consists of a set of data records or nodes linked together and organized by references (links or pointers). In linked data structures, the links are usually treated as special data types that can only be dereferenced or compared for equality. Linked data structures include linked lists, search trees, expression trees and many other widely used data structures.

This lab manual, *Lab: C and Data Structure*, contains several programs based on C concepts, such as If and Switch, Arrays, stack and queue, to provide the concept of programming. In addition, it will help students in coding and debugging their programs. The manual provides all logical, mathematical and conceptual programs that can help to write programs very easily in C language. These exercises shall be taken as the base reference during lab activities for students of BCA. There are also many Try Yourself Questions provided to students for implementation in the lab.

SYSTEM / SOFTWARE REQUIREMENTS

NOTES

You can use Turbo C++ compiler to run a C program only you need to save the source file with the .C extension instead of .CPP. Following are the system requirements to run a compolier.

1. Intel based desktop PC of 166MHz or faster processor with at least 64 MB RAM and 100 MB free disk space.
2. Turbo C++ compiler or GCC compilers

In this manual, we have used Turbo C++. To write C code first we need to open Turbo C++.

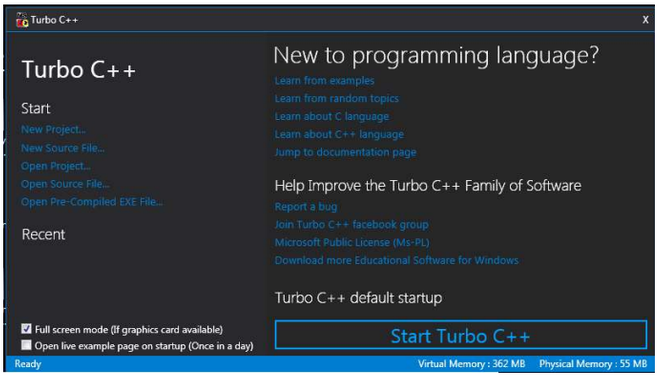
For every C program we need to follow following steps:

Write a program code → save your program (F2) → compile (Alt+F9) → Run(Ctrl +F9)

Step 1:

Click on Turbo C++ from start menu or double click on Turbo C++ on desktop

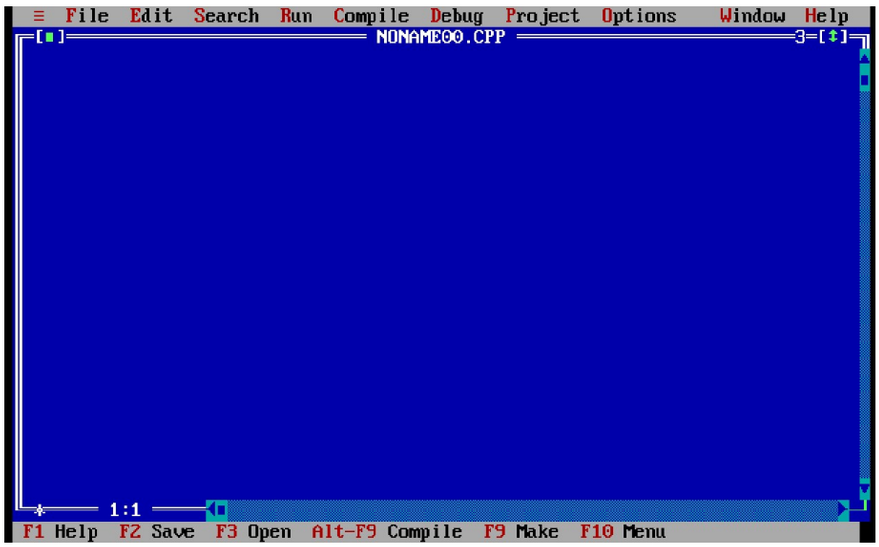
After clicking on Turbo C++ following screen will appear:



NOTES

Step 2:

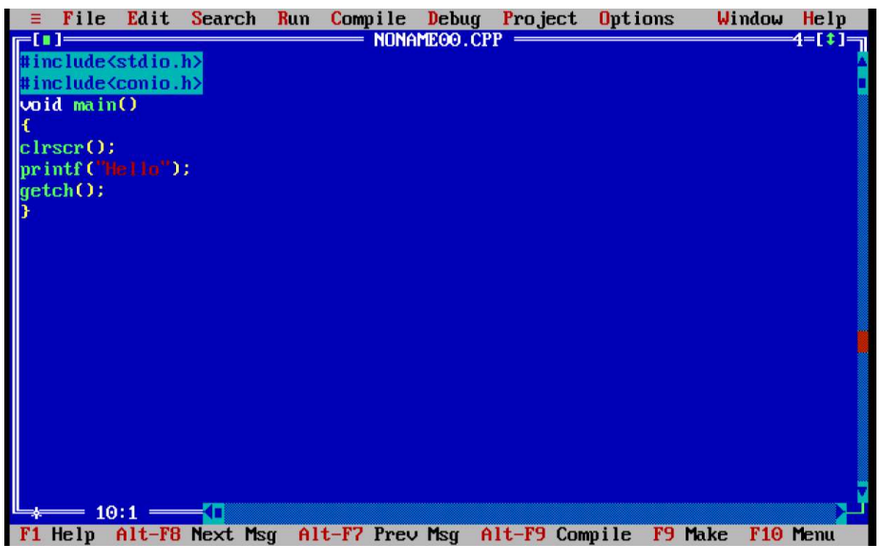
Click on Start Turbo C++. After clicking on Start Turbo C++ button following screen will appear:



This is the editor where we will write code of C programs.

Step 3:

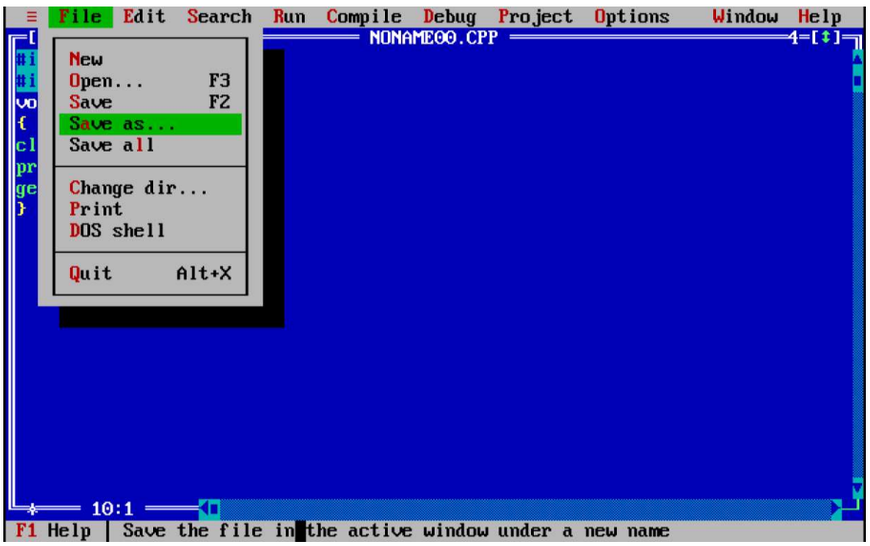
Write a program to print "Hello" on screen.



Step 4:

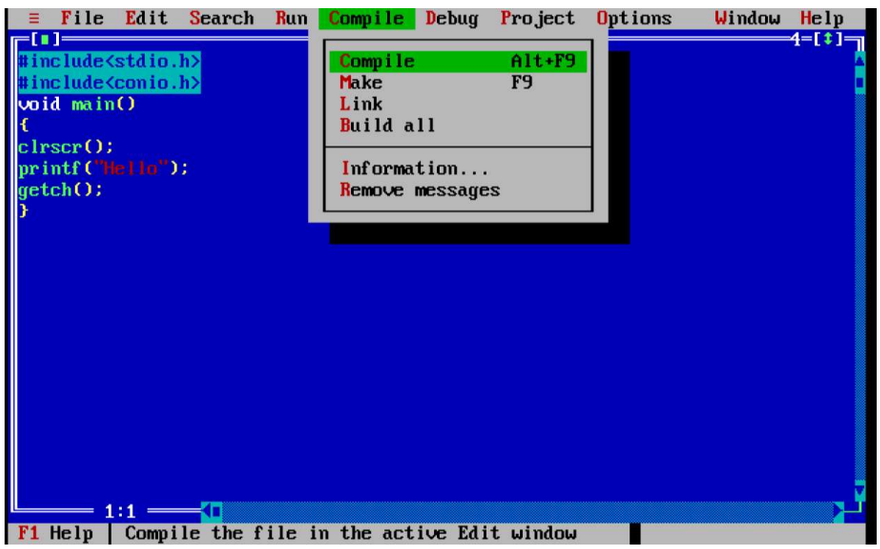
Save program by pressing F2 key or by using menu option File->Save As.

NOTES



Step 5:

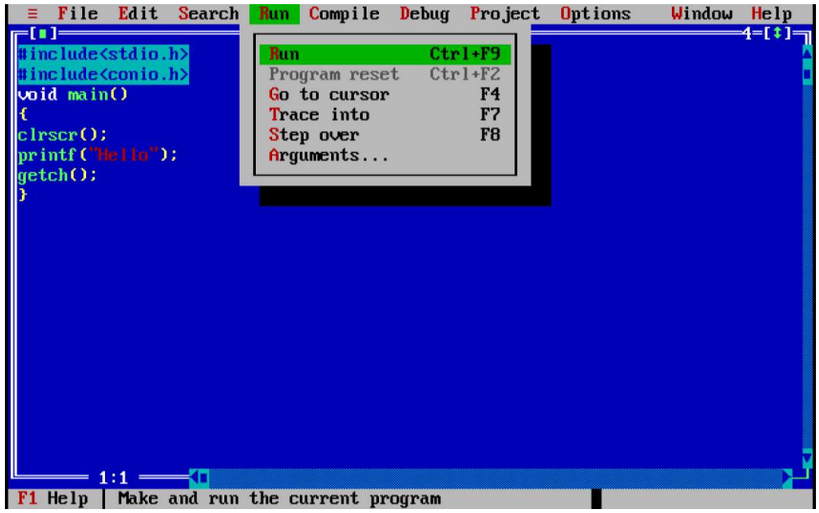
Compile program Hello.C by pressing Alt+F9 keys or by using menu option Compile-> Compile**.**



Step 6:

Run program Hello.C by pressing Ctrl +F9 keys or by using menu option Run-> Run.

NOTES



Output:

Programs:

1. Write a program to print sum and average of two given numbers.

```
#include<iostream.h>
#include<stdio.h>
 void main()
{
int num1,num2,sum,avg;

printf ("Enter two numbers"); //output statement
scanf ("%d",&num1); //input statement
scanf ("%d",&num2);

sum = num1+num2;
```

```
avg = sum/2;

printf ("Sum of two numbers %d", sum);
printf ("Average of two numbers %d" , avg);
}
```

NOTES

Output:

```
Enter two numbers
65
87
Sum of two numbers  152
Average of two numbers 76
```

2. Write a program to swap two numbers without using a third variable.

```
// Program to swap two numbers without using a third
variable
#include<iostream.h>
#include<stdio.h>
void main()
{
int num1,num2;
printf ("Enter two numbers"); //output statement
scanf ("%d",&num1); //input statement
scanf ("%d",&num2);
num1 = num2 - num1;
num2 = num2 - num1;

printf (" Values after swapping :\n");
printf ("\n Value of a Num1 %d" ,num1);
printf ("\n Value of a Num2 %d" ,num2);

}
```

Output:

Enter two numbers

3

4

Values after swapping :

Value of a num1 4

Value of a num2 3

NOTES

Try yourself:-

(i) Write a program to calculate volume of cylinder.

```
Volume of cylinder= PI*r*r*h
```

(ii) Write a program to calculate curved surface area of cylinder.

```
Curved surface area of cylinder= 2*PI*r*h
```

(iii) Write a program to print ASCII value of digits, uppercase and lowercase alphabets.

3. Write a program check whether the given number is even or odd.

```
#include<iostream.h>
#include <stdio.h>
void main()
{
int num;
printf ("Enter two numbers"); //output statement

scanf("%d",&num); //input statement
if (num%2==0)
{
 printf ("Number is even");

}
else
{
 printf("Number is odd");
}

}
```

Output:

```
Enter two number 1
Number is odd
```

4. Write a program to print the largest number among three numbers given by the user.

```
// program to print the largest number among three numbers

#include <stdio.h>
 void main()
{
int num1,num2,num3;
```

```
printf ("Enter three numbers");
scanf ("%d%d%d",&num1,&num2,&num3);


if (num1 >= num2 && num1 >= num3)
 {
 printf ("Largest number: %d",num1);

 }

 else if (num2 >= num1 && num2 >= num3)
 {
 printf ("Largest number: %d",num2);
 }

 else
 {
 printf ("Largest number: %d",num3);
 }

}
```

Output:

```
Enter three numbers
34
6
7
Largest number: 34
```

5. Write a program to print sum, difference, multiplication and division of two numbers according to the user choice using switch case.

```
#include<iostream.h>
#include <stdio.h>
void main()
{ int num1, num2; char op;
 printf ("Enter two numbers: ");
 scanf ("%d%d",&num1,&num2);
 printf ("Enter operator : ");
 scanf ("%c",&op);
 Switch (op)
 {
```

NOTES

NOTES

```
case '+': printf ("\n Sum of two numbers %d", num1+num2);
            break;
case '-': printf ("\n Subtraction of two numbers %d",
            num1-num2);
            break;
case '*': printf ("\n Multiplication of two numbers %d",
            num1*num2);
          break;
case '/': printf ("\n Division of two numbers %d", num1/
            num2);
          break;
default: printf ("\n Invalid operator");
            break;
 }
}
```

Output:

Enter two numbers: 56

9

Enter Operator: *

Multiplication of two numbers 504

Try yourself:-

(1) Write a program to convert a lowercase alphabet to uppercase or vice-versa.

(2) Write a program to check whether a year is leap year or not.

(3) Write a program to check whether a given character is uppercase or lowercase alphabet or a digit or a special character.

6. Write a program to print table of any number using for loop.

```
// program to print table of any number
#include <stdio.h>

void main()
{
 int num, i;
 printf ("Enter a numbers: ");
 scanf ("%d",&num);
 printf ("Table of %d\n",num);
for (i=1;i<=10;i++)
{
```

```
 printf ("%d\n",num*i);
}

}
```

NOTES

Output:

```
Enter a number: 9
Table of 9
9
18
27
36
45
54
63
72
81
90
```

7. Write a program to print Fibonacci Series (0, 1, 1, 2, 3, 5, 8, 13, 21...)

```
// Program to print Fibonacci series using for loop
#include <stdio.h>
#include<iostream.h>
void main()
{
 int num, i;
 printf("Enter a numbers: ");
 scanf("%d",&num);
 printf("Table of %d\n",num);
for(i=1;i<=10;i++)
{
 printf("%d\n",num*i);
}

}
```

Output:

```
Enter a number of terms for Series: 9
Fibonacci series :

0
1
1
2
3
5
8
13
21
```

NOTES

8. Write a program to check whether a given number is Armstrong.

A number is known as Armstrong number if sum of the cubes of its digits is equal to the number itself.

For example:

370 is an Armstrong number because:

$370 = 3^3 + 7^3 + 0^3$

$= 27 + 343 + 0$

$= 370$

```
// C Program to check Armstrong Number
#include<iostream.h>
#include <stdio.h>

void main()
{
 int num, sum = 0, rem, temp;
 printf ("Enter a number: ");
 scanf ("%d",&num);
 temp = num;
while (num>0)
{
rem = num%10;
sum = sum+(rem*rem*rem);
num = num/10;
}
if (temp==sum)
printf ("Given number is armstrong\n ");
else
printf ("Given number is not armstrong\n ");
}
```

Output:

Enter a number: 370

Given number is armstrong

9. Write a program to print table of any number using do while loop.

```
//C program to print table of any number using do while
loop
#include<iostream.h>
#include <stdio.h>
void main()
{
```

```
 int num, i;
 printf ("Enter a number: ");
 scanf ("%d",&num);
 printf ("\n Table of %d",num);
 i=1;
 do
 {
 printf ("\n %d",num*i);
 i++;
 }while(i<=10);
}
```

Output:

```
Enter any number: 12

 Table of12
12
24
36
48
60
72
84
96
108
120
```

Try yourself:-

(1) Write a program to reverse a number.

(2) Write a program to check whether a number is prime number or not.

(3) Write a program to convert binary number to decimal number.

10. Write a program to print factorial of a given number using recursive function.

```
//C Program to print factorial using recursive function
#include<stdio.h>
// Factorial Function
int factorial(int n)
{
 if (n > 1)
 return n * factorial(n - 1); //recursive call of factorial
                                    function
 else
 return 1;
}
```

NOTES

```
int main()
{
 int n;
 printf ("Enter a number : ");
 scanf ("%d",&n);

 printf ("Factorial of %d is %d", n, factorial(n));
 return 0;
}
```

Output:

11. Write a program to print Fibonacci series using recursion.

This C Program demonstrates the computation of Fibonacci Numbers using Recursion. For example Series 0, 1, 1, 2, 3, 5, 8, 13, 21 . . . . . N.

```
//C Program to print Fibonacci series using recursive
function
#include<stdio.h>
int fibonacci (int n)
{
 if ((n==1)||(n==0))
 {
 return (n);
 }
 else
 {

 return (fibonacci(n-1)+fibonacci(n-2));
                //recursive call of fibonacci function
 }
}

int main()
{
 int n, i;
```

```
 printf ("Enter number of terms for Fibonacci Series:");
 scanf ("%d", &n);
 printf ("Fibonacci Series \n");

for (i=0; i< n;i++)
 {
 printf (" %d ",fibonacci(i));
 }
return 0;
}
```

Output:

```
Enter number of terms for Fibonacci Series:12
Fibonacci Series
 0 1 1 2 3 5 8 13 21 34 55 89
```

Try yourself:-

(1) Write a program that uses a recursive function to find the binary equivalent of a given non-negative integer n.

(2) Write a programs functions to find the GCD of two given integers using recursive function.

12. Write a program to scan and print 5 values using array.

```
//C program to scan and print values using array
#include<iostream.h>
#include <stdio.h>
int main()

{

 int arr[5], i;
printf ("Enter 5 numbers:\n ");
 for (i=0;i<5;i++)
 scanf ("%d",&arr[i]);
 printf ("\n Array values are \n");
 for (i=0;i<5;i++)
 printf ("%d \n",arr[i]);
}
```

NOTES

NOTES

Output:

```
Enter 5 numbers:
 54
 2
 3
 54
 7

 Array values are
54
2
3
54
7
```

13. Write a program to print the largest value in an array.

```
//C program to print the largest value of an array
#include<iostream.h>
#include <stdio.h>
int main()
{

int arr[5],i,max;
 printf ("Enter 5 numbers:\n ");
 for (i=0;i<5;i++)
 scanf ("%d",&arr[i]);
max = arr[0];
 for (i = 1;i < 5; i++)
 {
 if (max < arr[i])
 max = arr[i];
 }
 printf ("Largest element = %d" ,max);
}
```

Output:

```
Enter 5 numbers:
 65
 4
 5
 76
 4
 Largest element = 76
```

14. Write a program that takes string as input and print it.

```
#include <stdio.h>
#include <conio.h>
```

NOTES

```
void main()
{
char str[15];
printf ("Enter your name: ");
scanf ("%s",str);
printf ("\nWelcome %s ",str);getch();
}
```

15. Write a program to print length of a given string without using string functions.

```
//C program to count string length
#include<stdio.h>

void main( )
{
int i,count=0;
 char str[50];
printf ("Enter any string ");

 gets (str); //gets function allows user to input string
with space

 //loop will run till it reaches to string terminator
'\0'
 for (i = 0; str[i] != '\0'; i++)
 {
 count++;
 }
 printf ("\n Length of string is %d", count);
}
```

Output:

```
Enter any string  programming

Length of string is  11
```

16. Write a program to check a string is palindrome or not.

```
#include<iostream.h>
#include<stdio.h>

int main( )
{
int i,len=0;
 char str[50],rev_str[50];
```

NOTES

```
printf ("Enter any string ");
gets (str); //gets function allows user to input string
              with space

//count length of string
for (i = 0; str[i] != '\0'; i++)
{
len++;
}

int j=0;
for (i = len - 1; i >= 0 ; i—,j++)
{
rev_str[j] = str[i];
}
   rev_str[j] ='\0'; //reverse string is terminated

//compare both strings
int flag=0;
for (i = 0; i < len ; i++)
{
if (str[i]==rev_str[i])
flag = 1;
else
{
break; //exit from loop
}
}

if (flag == 1)
   printf(" \nstring is a palindrome");
else
   printf(" \nstring is a not palindrome");}
```

Output:

```
Enter any string  nitin

string is a palindrome
```

Try yourself:-

(1) Write a program to insert an element in an array.

(2) Write a program to find sum of elements of an array.

(3) Write a program to find largest number from an array.

NOTES

17. Write a program to print sum of two matrices.

```
//C program to print sum of two matrices
#include<stdio.h>
int main()
{
 int i, j, m1[10][10], m2[10][10], sum[10][10];

 printf ("Enter the elements of first matrix\n");
 for ( i = 0 ;i < 3 ; i++ )
 {
 printf ("\n enter values for row %d \n",i+1);
 for ( j = 0 ; j<3 ; j++ )
 {
 scanf ("%d",&m1[i][j]);

 }
 }

 printf ("\n Enter the elements of second matrix\n");

 for ( i = 0 ;i < 3; i++ )
 {
 printf ("\n enter values for row %d \n",i+1);
 for ( j = 0 ; j< 3 ; j++ )
 {
 scanf ("%d",&m2[i][j]);
 }
 }
 printf ("Sum of two matrices \n");
 for ( i = 0 ;i < 3 ; i++ )
 {
 for ( j = 0 ; j<3 ; j++ )
 { sum[i][j] = m1[i][j]+m2[i][j];
 printf ("%d \t", sum[i][j]);
```

NOTES

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

}
```

Output:

```
Enter the elements of first matrix

 enter values for row  1
1 2 3

 enter values for row  2
2 3 4

 enter values for row  3
2 3 1
Enter the elements of second matrix

 enter values for row  1
5 4 3
 enter values for row  2
2 3 4

 enter values for row  3
4 5 6
Sum of two matrices
6   6   6
4   6   8
6   8   7
```

18. Write a program to perform matrix multiplication.

```
//C program for matrix multiplication
#include<stdio.h>

int main()
{

 int i,j,k, m1[10][10], m2[10][10], res[10][10];

 printf ("Enter the elements of first matrix\n");
 for ( i = 0 ;i < 3 ; i++ )
 {
 printf ("\n enter values for row %d \n",i+1);
 for ( j = 0 ; j<3 ; j++ )
 {
 scanf ("%d",&m1[i][j]);
```

```
 }

 }
 printf ("\nEnter the elements of second matrix\n");
 for ( i = 0 ;i < 3; i++ )
 {
 printf ("\n enter values for row %d \n",i+1);
 for ( j = 0 ; j< 3 ; j++ )
 {
 scanf ("%d",&m2[i][j]);
 }
 }
 for (i = 0; i < 3; ++i)
 {
 for (j = 0; j < 3; ++j)
 {
 res[i][j]=0;

 for (k = 0; k < 3; ++k)
 {
 res[i][j] += m1[i][k] * m2[k][j];
 }

 }
 }
printf ("Multiplication of two matrices \n");
 for ( i = 0 ;i < 3 ; i++ )
 {
 for ( j = 0 ; j<3 ; j++ )
 {

 printf ("%d \t",res[i][j]);

 }
 printf ("\n");
 }

 }
```

NOTES

Output:

NOTES

```
Enter the elements of first matrix

 enter values for row  1
1 2 3

 enter values for row  2
1 2 3

 enter values for row  3
1 2 3
Enter the elements of second matrix

 enter values for row  1
2 3 4

 enter values for row  2
2 3 4

 enter values for row  3
2 3 4
Multiplication of two matrices
12  18  24
12  18  24
12  18  24
```

Try yourself:-

(1) Write a program to print sum of diagonal values of a square Matrix.

(2) Write a program to find largest and smallest element of a Matrix.

(3) Write a program to convert first letter of each word of a string to uppercase and other to lowercase.

(4) Write a program to find substring in string (pattern matching).

File Handling Programs

File is a collection of bytes that is stored on secondary storage devices.

There are two types of files in a system:

1. Text files (ASCII)
2. Binary files

Text files contain ASCII codes of digits, alphabetic and symbols where as a Binary file contains collection of bytes (0's and 1's).

Basic Operations on File

1. Opening/Creating a file
2. Closing a file
3. Reading a file
4. Writing in a file

Various Mode of Operations on File

Mode	Description
r	Opens a file in read mode and returns null if file does not exist
w	Opens a file in write mode.
a	Opens a file in append mode
a+	Opens a file for read and write mode.

NOTES

19. Write a program to read name and roll number of n number of students from user and store them in a file.

```
#include <stdio.h>
int main()
{
 char name[50];
 int roll_no, i, num;

 printf ("Enter number of students: ");
 scanf ("%d", &num);

 File *fptr;
 fptr = (fopen("d:\\student.txt", "w"));
 if (fptr == NULL)
 {
 printf ("Can't open file");

 }

 for (i = 0; i < num; ++i)
 {
 printf ("\n student %d\n", i+1);
 printf ("Enter Roll number: ");
 scanf ("%d", &roll_no);

 printf ("Enter name: ");
 scanf ("%s", name);

  fprintf (fptr,"\nRoll number=%d \t Name: %s \n \n",
roll_no,name);
 }

 fclose (fptr);

}
```

Output:

NOTES

```
Enter number of students:  4

 student 1
Enter Roll nuber:  101
Enter name:  aaa

 student 2
Enter Roll nuber:  102
Enter name:  bbb

 student 3
Enter Roll nuber:  103
Enter name:  ccc

 student 4
Enter Roll nuber:  104
Enter name:  ddd
```

20. Write a program to print values of an array using pointers.

```
#include<iostream.h>
#include <stdio.h>

void main()
{
 int arr[5],i,*ptr;
 ptr = arr;     //ptr pointer is holding address of arr[0]
                      element.
 printf ("Enter 5 numbers:\n ");
 for (i=0;i<5;i++)
 scanf ("%d",&arr[i]);
printf ("\n Array values using pointer \n");
 for (i=0;i<5;i++)
 printf ("%d\n",*(ptr + i)); //*(ptr+i) will print array
                                  values

}
```

21. Write a program to input and print student data using structure.

```
//C program to input and print student data using structure
#include<stdio.h>
//student structure
 struct student
 {
 int rno;
 char name[10];
 };
```

NOTES

```
int main()
{
 struct student obj;
 //input values
 printf ("\n enter student roll number :");
scanf ("%d",&obj.rno);
printf ("\n enter student name :");
scanf ("%s",&obj.name);

//display values
 printf ("\n roll number : %d ",obj.rno);
 printf ("\n Name: %s", obj.name);

 }
```

Output:

```
enter student roll number : 22

enter student name : dhruv

roll number : 22
Name: dhruv
```

22. Write a program to input and print student data using structure array.

```
//C program to input and print student data using structure
array

#include<stdio.h>

//student structure
 struct student
 {
 int rno;
 char name[10];
 };

int main()
{
 int i;
 struct student obj[5]; // structure object as an array
```

NOTES

```
printf ("\n enter student roll number and name :");
for (i=0;i<5;i++)
{
printf ("\n Roll number :");
scanf ("%d",&obj[i].rno);
printf ("\n Name :");
scanf ("%s",&obj[i].name);

}
//display values
printf ("\n student roll number and name \n");

for (i=0;i<5;i++)
{
printf ("\n Roll number\t: %d ",obj[i].rno);
printf ("\n Name \t: %s", obj[i].name);
}
}
```

Output:

```
enter student roll number and name :
Roll number : 1

Name : dhruv

Roll number : 2

Name : aakarsh

Roll number : 3

Name : tarush

Roll number : 4

Name : sarthak

Roll number : 5

Name : siddharth

 student roll number and name

Roll number    : 1
Name   : dhruv
Roll number    : 2
Name   : aakarsh
Roll number    : 3
Name   : tarush
Roll number    : 4
Name   : sarthak
Roll number    : 5
Name   : siddharth
```

23. Write a program to input and print student data using pointer object of structure.

NOTES

```
//C program to input and print student data using pointer
object of structure.
#include<iostream.h>
#include<stdio.h>
//student structure
 struct student
 {
 int rno;
 char name[10];
 };

int main()
{
 struct student obj,*ptr;
 //input values
 printf ("\n enter student roll number :");
scanf ("%d",&obj.rno);
printf ("\n enter student name :");
scanf ("%s",&obj.name);

//assigning address of structure object to pointer
ptr=&obj;
//display values
 printf ("\n roll number : %d ",ptr->rno);
 printf ("\n Name: %s", ptr->name);

 }
```

Output:

```
enter student roll number : 11

enter student name : aakarsh

roll number : 11
Name: aakarsh
```

Union is a user defined data type similar to structure but in union, all members share the same memory location.

In the example given below, both a and b share the same location, if we change a, we can see the changes being reflected in b.

NOTES

24. Write a program to illustrate the concept of union.

```
//C program to demonstrate union
#include<iostream.h>
#include <stdio.h>

// union
union number
{
 int a, b;
};

void main()
{
 // A union object obj
 union number obj;
 obj.a=10;
 printf ("value of a and b after initializing value to a only\n");
 printf ("value of a and b = %d \n value of b = %d",obj.a, obj.b);

 printf ("\n\n value of a and b after initializing value to b only");
 obj.b=200;
 printf ("\nvalue of a = %d \n value of b = %d",obj.a, obj.b);

}
```

Output:

```
value of a and b after initializing value to a only
value of a and b = 10
 value of b = 10

 value of a and b after initializing value to b only
value of a = 200
 value of b = 200
```

Stack is a linear data structure which has LIFO (Last in First Out) property. It contains only one pointer known as TOP that points top most element of Stack. In stack insertion and deletion is allowed only from top. Where insertion in stack is also known as a PUSH operation and deletion from stack is also known as POP operation in stack.

NOTES

Stack Implementation

Stack can be implemented in two ways:

(i) Stack implementation using array.

(ii) Stack implementation using linked list.

Operations on Stack:

(i) Push ( ) operation insert an item to the top of stack.

(ii) Pop ( ) operation deletes an item from the top of stack

Applications of Stack

(i) Conversion of infix expression into Prefix expression

(ii) Conversion of infix expression into Postfix expression

(iii) Evaluation of Postfix expression

(iv) Tower or Hanoi

(v) Recursion

Array Implementation of Stack

Algorithm: To PUSH value in stack using array.

Description:

Here Stack is an array where we will PUSH and POP values.

MAX is a constant used to define maximum limit of

TOP is the top most element of Stack where insertion and deletion is allowed

DATA is data to be inserted in Stack

PUSH (Stack, Top, Data)

Algorithm to push an item on to stack.

1. IF TOP = MAX then
2. Write: "Stack is overflow"
3. Exit
4. Else

   `[increment TOP by 1]`
5. Set TOP = TOP + 1
6. Set STACK [TOP]= DATA

   `[End of IF STRUCTURE]`
7. END

Algorithm: To POP value from stack using array.

Description:

NOTES

Here Stack is an array where we will PUSH and POP values.

TOP is the top most element of Stack where insertion and deletion is allowed

DATA is deleted data

POP (Stack, Top)

Algorithm to push an item into stack.

1. IF TOP = 0 then
2. Write: "Stack is underflow"
3. Exit
4. Else
5. Set DATA = STACK [TOP]
   [decrease TOP by 1]
6. Set TOP = TOP- 1
   [End of IF STRUCTURE]
7. END

25. Write a C Program to implement Stack using Array.

```
#include <stdio.h>
#define MAX 5

 int top=-1;
 int ele[MAX];

//PUSH function

void push( int item )
{
 if ( top==MAX-1 )
 {
 printf ("Stack Overflow \n");
 }
else
{

 top++;
 ele [top] = item;
```

```
 printf ("\n Inserted value is : %d",item);
}
}

 //POP function
int pop( )
{
 int item;
 if ( top==-1 )
 {
 printf ("\nStack Underflow");
 }
 else
{
 item = ele[top];

top—;
}
return item;
 }

//DISPLAY function
void display ()
{
 if ( top==-1 )
 {
 printf ("\nStack Underflow");
 }
 else
{
 int i;
 printf ("Stack value are \n");

for (i=top;i>=0;i—)
printf ("%d \n",ele[i]);
}
 }
```

NOTES

NOTES

```
void main ()
{
int item = 0, choice, value; char ans;

 do
{

 printf ("1. Push \n2. Pop\n3. Display\n4. Exit\n");
 printf ("\nEnter your choice: ");
 scanf ("%d",&choice);

 switch (choice)
 {
 case 1:
    printf ("Enter the value to be insert: ");
    scanf ("%d",&value);
    push (value);
    break;
 case 2:
    value = pop();
     printf ("\nDeleted value is %d ",value);
    break;
 case 3:
    display ();
     break;
 case 4:
    exit(0);
 default:
    printf ("Invalid choice");
 }

 printf ("\nDo you want to cont…(y/n)");
 scanf ("%s",&ans);
 }while (ans=='y' || ans=='Y');

}
```

Output:

```
1. Push"<<endl<<"2. Pop"<<endl<<"3. Display"<<endl<<"4. Exit
Enter your choice: 1
Enter the value to be insert: 11

 Inserted value is : 11
Do you want to cont…(y/n)y1. Push"<<endl<<"2. Pop"<<endl<<"3. Display"<<endl<<"4. Exit
Enter your choice:
1
Enter the value to be insert: 22

 Inserted value is : 22
Do you want to cont…(y/n)y
1. Push"<<endl<<"2. Pop"<<endl<<"3. Display"<<endl<<"4. Exit
Enter your choice: 3
Stack value are
22
11

Do you want to cont…(y/n)y1. Push"<<endl<<"2. Pop"<<endl<<"3. Display"<<endl<<"4. Exit
Enter your choice:
2

Deleted value is 22
Do you want to cont…(y/n)y
1. Push"<<endl<<"2. Pop"<<endl<<"3. Display"<<endl<<"4. Exit
Enter your choice: 3
Stack value are
11

Do you want to cont…(y/n)n
```

Algorithm: To transform infix expression to postfix expression value from stack using array.

Description:

- Here Q is an arithmetic expression written in infix expression.
- Algorithm finds P as a postfix expression
- Stack is an array where we will PUSH and POP values.
- MAX is a constant used to define maximum limit of
- TOP is the top most element of Stack where insertion and deletion is allowed
- DATA is data to be inserted in Stack.

1. Push "(" onto Stack, and add ")" to the end of Q.
2. Scan Q from left to right and repeat Step 3 to 6 for each element of Q until the Stack is empty.
3. If an operand is encountered, add it to P.
4. If a "(" left parenthesis is encountered THEN push it onto Stack.
5. If an operator is encountered ,then:
   (i) Repeatedly POP from Stack and add to P each operator (on the top of Stack) which has the same precedence as or higher precedence than operator.
   (ii) Add operator to Stack.

   [End of If]

NOTES

NOTES

6. If a right parenthesis is encountered ,then:
   (i) Repeatedly pop from Stack and add to P each operator (on the top of Stack) until a left parenthesis is encountered.
   (ii) Remove the left Parenthesis.

```
    [End of If]
    [End of If]
```

7. END.

26. Write a program to convert infix expression to postfix expression using stack.

```
//C program to convert infix expression to postfix
expression
#include<stdio.h>
#include <ctype.h>
#define SIZE 50

char s[SIZE];
int top=-1;

 //push Function
void push(char elem)
{
 s[++top]=elem;
}

 //pop function
char pop()
{
 return (s[top—]);
}

 // precedence function
int pr(char elem)
{
 switch (elem)
 {
 case '#': return 0;
 case '(': return 1;
 case '+':
 case '-': return 2;
 case '*':
 case '/': return 3;
```

NOTES

```
 }
}
 //main function
void main()
{
 char infx[50],pofx[50],ch,elem;
 int i=0,k=0;
 printf ("\n\n Enter the Infix Expression (Ex. A+B-2) ");
 scanf ("%s",infx);
 push ('#');
 while ( (ch=infx[i++]) != '\0')
 {
 if ( ch == '(') push(ch);
 else
 if (isalnum(ch)) pofx[k++]=ch;
 else
 if ( ch == ')')
 {
 while ( s[top] != '(')
 pofx [k++]=pop();
 elem = pop(); /* Remove ( */
 }
 else
 { /* Operator */
 while ( pr(s[top]) >= pr(ch) )
 pofx [k++]=pop();
 push (ch);
 }
 }
 // Pop from stack till empty
 while ( s[top] != '#')
 pofx [k++]=pop();
 pofx [k]='\0';
 printf ("\n\n Infix Expn: %s \n Postfix Expn: %s\n",
infx, pofx);
}
```

Output:

```
Enter the Infix Expression (Ex. A+B-2)  3+4/2


 Infix Expn: 3+4/2
Postfix Expn: 342/+
```

NOTES

Algorithm: To evaluate postfix expression using stack.

Description:

- Here P is an arithmetic expression written in postfix expression.
- Stack is an array where we will PUSH all the operands and final value.
- TOP is the top most element of Stack where insertion and deletion is allowed

1. Add a Right Parenthesis ")" at the end of P postfix expression.
2. Scan P from left to right and repeat Step 3 to 4 for each element of P until ")" is encountered.
3. If an operand is encountered, add it to stack.
4. If an operator * is encountered, then
   (i) Remove two TOP elements of the stack where A is the TOP and B is the TOP-1 element.
   (ii) Evaluate B * A.
   (iii) Place the result of step (b) on to stack.
5. Set value equal to TOP element on stack.
6. Exit

27. Write program to evaluation of postfix expressions using stack.

```
//C program to evaluate postfix expressions using stack.
#include<stdio.h>
#include<ctype.h>

int stack[20];
int top = -1;

void push(int x)
{
 stack [++top] = x;
}

int pop()
{
 return stack[top—];
}

int main()
{
 char exp[20];
```

```
char *e;
int n1,n2,n3,num;
printf ("Enter the expression :: ");
scanf ("%s",exp);
e = exp;
while (*e != '\0')
{
if (isdigit(*e))
{
num = *e - 48;
push (num);
}
else
{
n1 = pop ();
n2 = pop ();
switch (*e)
{
case '+':
{
n3 = n1 + n2;
break;
}
case '-':
{
n3 = n2 - n1;
break;
}
case '*':
{
n3 = n1 * n2;
break;
}
case '/':
{
n3 = n2 / n1;
break;
}
}
push (n3);
```

NOTES

NOTES

```
    }
    e++;
    }
    printf ("\n The result of expression %s = %d\n\n", exp,
    pop());

    }
```

Output:

```
Enter the expression ::  35-3*

The result of expression 35-3*  =  -6
```

Algorithm: To push value in stack using linked list.

Description:

- Here TOP is a pointer variable which contains the address of TOP node.
- NEW is the new node to be inserted in linked list.
- INFO is data to be inserted in linked list.

Push_Stack (Top, Info) [Overflow?]

```
1  IF NEW = NULL THEN
 WRITE: OVERFLOW
 EXIT
2 ELSE
IF TOP = NULL THEN
SET NEXT [TOP] = NULL

ELSE
SET NEXT [NEW] = TOP
[END OF IF]
SET DATA [TOP] = INFO
SET TOP = NEW

[END OF IF]
3 END
```

Algorithm: To pop value from stack using linked list.

Description:

- Here TOP is the Top node of Stack to be deleted.
- TEMP is name given to the node to be deleted from the list.

POP_STACK (TOP)

1. [UNDERFLOW?]
   IF TOP = NULL THEN
   WRITE: UNDER FLOW
   EXIT
2. ELSE
   SET INFO = DATA [TOP]
   SET TEMP = TOP
   SET TOP =NEXT [TOP]
   DELETE TEMP
   [END OF IF]
3. END

NOTES

28. Write a program for stack implementation using linked list.

```
//C program for stack implementation using linked list

#include <stdio.h>
#include <stdlib.h>

struct node
{
 int info;
 struct node *ptr;
}*top,*curr,*temp;


//Push data into stack
void push(int data)
{
 if (top == NULL)
 {
 top =(struct node *)malloc(1*sizeof(struct node));
 top->ptr = NULL;
 top->info = data;
 }
 else
 {
 temp =(struct node *)malloc(1*sizeof(struct node));
 temp->ptr = top;
 temp->info = data;
 top = temp;
```

NOTES

```
 }

}

 // Pop Operation on stack
void pop()
{
 curr = top;

 if (curr == NULL)
 {
 printf ("\nStack is empty");
 return;
 }
 else
 curr = curr->ptr;
 printf ("\n Popped value : %d", top->info);
 free(top);
 top = curr;

}


// Display stack elements
void display()
{
 curr = top;

 if (curr == NULL)
 {
 printf ("Stack is empty");
 return;
 }

 while (curr != NULL)
 {
 printf ("%d ", curr->info);
 curr = curr->ptr;
 }
 }
```

NOTES

```
void main()
{
 int no, ch, e;
 char ans;
 top = NULL;

 printf ("\n 1. Push \n 2. Pop \n 3. Print \n 4. Exit
\n");



 do
 {
 printf ("\n Enter choice : ");
 scanf ("%d", &ch);

 switch (ch)
 {
 case 1:
 printf ("Enter data : ");
 scanf ("%d", &no);
 push (no);
 break;
 case 2:
 pop ();
 break;
 case 3:
 display ();
 break;
 case 4:
 exit(0);

 default: printf(" Invalid choice ");
 }
 printf ("\ndo you want to cont...(Y/N)");
 scanf ("%s",&ans);
 } while (ans=='y' ||ans=='Y');

 }
```

Output:

NOTES

```
 1. Push
 2. Pop
 3. Print
 4. Exit

 Enter choice :  1
Enter data :  11

do you want to cont...(Y/N) y

 Enter choice :  1
Enter data :  22

do you want to cont...(Y/N) y

 Enter choice :  3
22 11
do you want to cont...(Y/N) y

 Enter choice :  2

 Popped value : 22
do you want to cont...(Y/N) n
```

Try yourself:-

(1) Write a program to solve Towers of Hanoi problem using a recursive function.

(2) Write a program to convert infix expression to prefix expression.

Algorithm: For insertion in queue using array.

```
INS/ERT_Queue (INT Data)
IF (REAR=SIZE) THEN
WRITE "QUEUE IS FULL"
ELSE IF (REAR=0) THEN
FRONT=1
END IF
REAR=REAR+1
QUEUE [REAR] =DATA
EXIT
```

Algorithm: For deletion in queue using array.

```
DELETE-QUEUE ( )

IF (FRONT=0) THEN
WRITE "QUEUE IS EMPTY"
ELSE
ITEM=QUEUE [FRONT]
IF (FRONT=REAR) THEN
REAR=0
FRONT=0
ELSE
```

```
FRONT=FRONT+1
END IF
END IF
EXIT
```

29. Write a program to implement queue using array.

```
//Program to implement queue using array
#include <stdio.h>

#define MAX 50

int queue[MAX];
int rear = - 1;
int front = - 1;

//insert or enqueue function
void enqueue (int data)
{

 if (rear == MAX - 1)
 printf ("Queue Overflow \n");
 else
 {
 if (front == - 1)
 front = 0;

 rear = rear + 1;
 queue [rear] = data;
 }
}

//dequeue or delete function
void dequeue()
{
 int data;
 if (front == - 1 || front > rear)
 {
 printf ("Queue Underflow \n");
 }
 else if (front == rear)
 {
 data = queue[front];
```

NOTES

NOTES

```
 front = rear=-1;
 }

 else
 {
 data = queue[front];
 front = front + 1;
 }

 printf ("Deleted value is %d\n",data);
}

//display function
void display()
{
 int i;
 if (front == - 1)
 printf ("Queue is empty \n");
 else
 {
 printf ("Queue is : \n");
 for (i = front; i <= rear; i++)
 printf ("%d ", queue[i]);
 printf ("\n");
 }
}

//main function
void main()
{
 int choice, item;
 char ans;
 do
 {

 printf ("1.Insert \n 2.Delete \n3.Display \n");
 printf ("Enter your choice : ");
 scanf ("%d", &choice);
```

```
switch (choice)
{
case 1: printf("\nEnter value to be inserted : ");
          scanf("%d", &item);
          enqueue (item);
            break;
case 2:      dequeue ();
             break;
case 3:   display();
          break;
default: printf("Invalid choice \n");
         break;
}
printf("do you want to cont...(Y/N)");
scanf("%s",&ans);
} while (ans=='y' ||ans=='Y');

}
```

Output:

```
1.Insert
 2.Delete
3.Display
Enter your choice :  1

Enter value to be inserted  :  11
do you want to cont...(Y/N) y
1.Insert
 2.Delete
3.Display
Enter your choice :  1

Enter value to be inserted  :  22
do you want to cont...(Y/N) y
1.Insert
 2.Delete
3.Display
Enter your choice :  33
Invalid choice
do you want to cont...(Y/N) y
1.Insert
 2.Delete
3.Display
Enter your choice :  1

Enter value to be inserted  :  33
do you want to cont...(Y/N) y
1.Insert
 2.Delete
3.Display
Enter your choice :  3
```

NOTES

30. Write a program to implement circular queue using array.

NOTES

```
//Program to implement queue using array
#include <stdio.h>

#define MAX 5

int queue[MAX];
int rear = - 1;
int front = - 1;

//insert or enqueue function
void cqinsertion (int data)
{
 //circular queue is full or not
 if (front==(rear+1) %MAX)
 {

 printf ("Queue Overflow \n");
 }
 else
 {
 //element is the first element?
 if (front == - 1)
 {
 front = rear=0;
 }
 else
 {
 rear =(rear+1) %MAX;
 }

 queue [rear] = data;
 }
}

//dequeue or delete function
void cqdeletion()
{
 int data;
 if (front == - 1)
```

```
 {
 printf ("Queue Underflow \n");
 }
 else
 {
 data = queue[front];

 }

//check if deleted value is the last element
 if (front==rear)
 {
 front =rear=0;
 }
 else
 {
 front ==(front+1) %MAX;
 }

 printf ("Deleted value is %d\n", data);
}

//display function
void display()
{
 int i;
 if (front == - 1)
 {
 printf ("Queue is empty \n");
 }
 else
 {
 printf ("Queue is : \n");
 for (i = front; i <= rear; i++)
 {
 printf ("%d \n ", queue[i]);
 }

 }
 if (front>rear)
```

NOTES

NOTES

```
 {
 for (i=front;i<MAX;i++)
 printf ("%d \n ", queue[i]);

 for (i=0;i<rear;i++)
 printf ("%d \n ", queue[i]);

 }
}

//main function
void main()
{
 int choice, item;
 char ans;
 do
 {

 printf ("1.Insert \n 2.Delete \n3.Display \n");
 printf ("Enter your choice : ");
 scanf ("%d", &choice);
 switch (choice)
 {
 case 1: printf("\n Enter value to be inserted : ");
          scanf ("%d", &item);
       cqinsertion (item);
       break;
 case 2: cqdeletion ();
          break;
 case 3: display();
         break;
 default: printf("Invalid choice \n");
         break;
 }
 printf("do you want to cont...(Y/N)");
 scanf("%s",&ans);
 } while (ans=='y' ||ans=='Y');

}
```

Output:

```
1.Insert
 2.Delete
3.Display
Enter your choice :  1

Enter value to be inserted  :  11
do you want to cont...(Y/N) y
1.Insert
 2.Delete
3.Display
Enter your choice :  1

Enter value to be inserted  :  22
do you want to cont...(Y/N) y
1.Insert
 2.Delete
3.Display
Enter your choice :  1

Enter value to be inserted  :  33
do you want to cont...(Y/N) y
1.Insert
 2.Delete
3.Display
Enter your choice :  44
invalid choice
do you want to cont...(Y/N) y
1.Insert
 2.Delete
3.Display
Enter your choice :  1

Enter value to be inserted  :  55
do you want to cont...(Y/N) y
1.Insert
 2.Delete
3.Display
Enter your choice :  3
Queue is :
11
```

NOTES

31. Write a program to implement queue using linked list.

```
//C program Queue implementation using linked list
#include <stdio.h>
#include <stdlib.h>

struct node
{
 int info;
 struct node *next;
}*front,*rear,*temp,*curr;

// Enqueue or insert element in queue
void enqueue(int data)
{
 if (rear == NULL)
 {
 rear = (struct node *)malloc(1*sizeof(struct node));
 rear->next = NULL;
 rear->info = data;
```

NOTES

```
 front = rear;
 }
 else
 {
 temp =(struct node *)malloc(1*sizeof(struct node));
 rear->next = temp;
 temp->info = data;
 temp->next = NULL;
 rear = temp;
 }
}

 // Dequeue or delete element
void dequeue()
{
 curr = front;

 if (curr == NULL)
 {
 printf ("\n Error: Trying to display elements from empty
queue");
 return;
 }
 else
 if (curr->next != NULL)
 {
 curr = curr->next;
 printf("\n Dequed value : %d", front->info);
 free(front);
 front = curr;
 }
 else
 {
 printf ("\n Dequed value : %d", front->info);
 free (front);
 front = NULL;
 rear = NULL;
 }

}
```

```
// Displaying queue
void display()
{
 curr = front;

 if ((curr == NULL) && (rear == NULL))
 {
 printf ("Queue is empty");

 }
 while (curr != rear)
 {
 printf ("%d ", curr->info);
 curr = curr->next;
 }
 if (curr == rear)
 printf ("%d", curr->info);
}


void main()
{
 int data, ch, e;
 char ans;
 front = rear = NULL;

 do
 {
 printf ("\n 1. Insert \n 2. Delete \n 3. Display \n 4.
Exit \n");

 printf ("\n Enter choice : ");
 scanf ("%d", &ch);
 switch (ch)
 {
 case 1:
    printf ("Enter data : ");
    scanf ("%d", &data);
```

NOTES

NOTES

```
        enqueue (data);
      break;
   case 2:
       dequeue ();
      break;
   case 3:
      display ();
      break;
   case 4:
       exit(0);

   default :
   printf (" Invalid choice ");
         break;
   }
   printf ("\n do you want to cont...(Y/N)");
   scanf ("%s", &ans);
   } while (ans=='y' || ans=='Y');
  }
```

Output:

```
 1. Insert
 2. Delete
 3. Display
 4. Exit

 Enter choice :  1
Enter data :  11

do you want to cont...(Y/N) y

 1. Insert
 2. Delete
 3. Display
 4. Exit

 Enter choice :  1
Enter data :  22

do you want to cont...(Y/N) y

 1. Insert
 2. Delete
 3. Display
 4. Exit

 Enter choice :  3
11 22
do you want to cont...(Y/N) y

 1. Insert
 2. Delete
 3. Display
 4. Exit

 Enter choice :  2
```

Try yourself:-

(1) Write a program to implement circular queue using linked list.

(2) Write a program to implement priority queue.

NOTES

32. Write a program to insert node at the beginning of single linked list.

```
//C program to insert node at the beginning of single
linked list
#include <stdio.h>
#include <stdlib.h>

//structure for node
struct node
{
 int data;
 struct node *next;
}*start,*end,*temp,*curr;

//Function to insert at the beginning

 void insert_beginning (int d)
 {
 //allocate memory to new node
 temp = (struct node *)malloc(1*sizeof(struct node));
 temp->data = d;
 temp->next = NULL;

 if (start == NULL)
 {
 start =end= temp;
 end->next=NULL;
 }
 else
 {
 temp->next = start;
 start=temp;
 }
 }
```

NOTES

```
//Function to traversal/print single linked list
 void traversal()
 {
 if (start == NULL)
 {
 printf ("Underflow \n");
 }
 else
 {
 curr =start;
 while (curr!=NULL)
 {
 printf ("%d \n ",curr->data);

 curr = curr->next;
 }

 }
 }

void main()
{
 int d, ch;
 char ans;
 start = end= NULL;
 do
 {
 printf ("\nEnter value to be inserted ");
 scanf ("%d",&d);
 insert_beginning (d);
 printf ("\n\nCont...(y/n)");
 scanf ("%s",&ans);
 }while (ans=='y'|| ans=='Y');

 printf ("\n Value are \n");
 traversal ();

}
```

Output:

```
Enter value to be inserted  23

Cont...(y/n) y

Enter value to be inserted  43

Cont...(y/n) y

Enter value to be inserted  34

Cont...(y/n) n

 Value are
34
 43
 23
```

NOTES

33. Write a program to insert node at the end of single linked list.

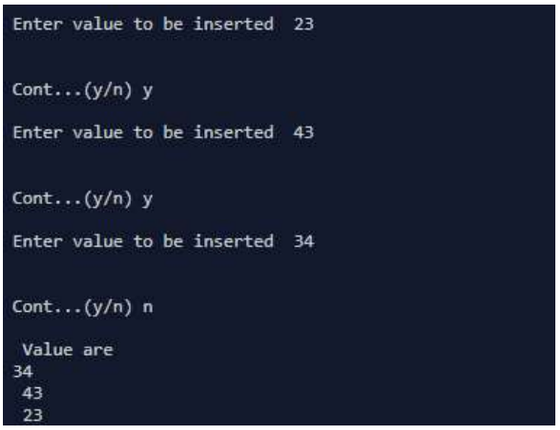
```
//C program to insert node at the end of single linked
list
#include <stdio.h>
#include <stdlib.h>

//structure for node
struct node
{
 int data;
 struct node *next;
}*start,*end,*temp,*curr;

//Function to insert at the end

 void insert_end(int d)
 {


 //allocate memory to new node
 temp = (struct node *)malloc(1*sizeof(struct node));
 temp->data = d;
 temp->next = NULL;
```

NOTES

```
 if (start == NULL)
 {
 start =end= temp;
 end->next=NULL;
 }
 else
 {
 temp->next = start;
 start=temp;
 }
 }


//Function to traversal/print single linked list
 void traversal()
 {
 if (start == NULL)
 {
 printf ("Underflow \n");
 }
 else
 {
 curr =start;
 while (curr!=NULL)
 {
 printf ("%d \n ",curr->data);

 curr =curr->next;
 }

 }
 }

void main()
{
 int d, ch;
 char ans;
 start = end= NULL;
```

```
do
{
printf ("\nEnter value to be inserted ");
scanf ("%d",&d);
insert_beginning (d);
printf ("\n\nCont...(y/n)");
scanf ("%s",&ans);
}while (ans=='y'|| ans=='Y');

printf("\n Value are \n");
traversal ();

}
```

Output:



34. Write a program to insert node at the beginning of double linked list.

```
//C program to Insert Node at the Beginning of double
linked list
#include <stdio.h>
#include <stdlib.h>

//structure for node

struct node
{
 struct node *prev;
 int data;
 struct node *next;
}*start,*end,*temp,*curr;
```

NOTES

```
//Function to insert at the beginning in double linked
list

 void insert_beginning (int d)
 {
 //allocate memory to new node
 temp = (struct node *) malloc (1*sizeof (struct node));
 temp->data = d;
 temp->next = NULL;

 if (start == NULL)
 {
 start =end= temp;
 end->next=NULL;
 end->prev=NULL;

 }
 else
 {
 temp->prev=NULL;
 temp->next=start;
 start->prev=temp;
 start = temp;
 }



 }

 //Function to traversal/print double linked list (START
to END)
 void traversal_S_to_E()
 {

 if (start == NULL)
 {
 printf ("Underflow \n");
 }
 else
 {
 curr =start;
```

```
 while (curr!=NULL)
 {
 printf ("%d \n ",curr->data);

 curr =curr->next;
 }

 }
 }

//Function to traversal/print double linked list (END to
START)
 void traversal_E_to_S()
 {
 if (end == NULL)
 {
 printf ("Underflow \n");
 }
 else
 {
 curr=end;
 while (curr!=NULL)
 {
 Printf ("%d \n ",curr->data);

 curr =curr->prev;
 }

 }
 }


void main()
{
 int d, ch;
 char ans;
 start = end= NULL;
 do
 {
 printf ("\nEnter value to be inserted ");
```

NOTES

NOTES

```
scanf ("%d",&d);
insert_beginning (d);
printf ("\n\nCont...(y/n)");
scanf ("%s",&ans);
}while (ans=='y'|| ans=='Y');


printf ("\n Value of list from start to end \n");
traversal_S_to_E ();
printf ("\n Value of list from end to start \n");
traversal_E_to_S ();

}
```

Output:

```
Enter value to be inserted  22

Cont...(y/n) y

Enter value to be inserted  55

Cont...(y/n) y

Enter value to be inserted  65

Cont...(y/n) n

 Value of list from start to end
65
 55
 22

 Value of list from end to start
22
 55
 65
```

35. Write a program to insert node at the end of double linked list.

```
//C program to insert node at the end of double linked
list

#include <stdio.h>
#include <stdlib.h>

//structure for node

struct node
{
 struct node *prev;
```

```
 int data;
 struct node *next;
}*start,*end,*temp,*curr;

//Function to insert at the end in double linked list
 void insert_end (int d)
 {

 //allocate memory to new node
 temp = (struct node *)malloc(1*sizeof(struct node));
 temp->data = d;
 temp->next = NULL;

 if (start == NULL)
 {
 start =end= temp;
 end->next=NULL;
 end->prev=NULL;

 }
 else
 {
 end->next=temp;
 temp->prev=end;
 temp->next=NULL;
 end =temp;
 }

}

 //Function to traversal/print double linked list (START
to END)
 void traversal_S_to_E()
 {

 if (start == NULL)
 {
 printf ("Underflow \n");
 }
 else
```

NOTES

NOTES

```
 {
 curr =start;
 while (curr!=NULL)
 {
 printf ("%d \n ",curr->data);

 curr = curr->next;
 }

 }
 }

//Function to traversal/print double linked list (END to
START)
 void traversal_E_to_S()
 {
 if (end == NULL)
 {
 printf ("Underflow \n");
 }
 else
 {
 curr = end;
 while (curr!=NULL)
 {
 printf ("%d \n ", curr->data);

 curr = curr->prev;
 }

 }
 }


void main()
{
 int d, ch;
 char ans;
 start = end= NULL;
 do
 {
```

```
printf ("\n Enter value to be inserted ");
scanf ("%d",&d);
insert_end (d);
printf ("\n\nCont...(y/n)");
scanf ("%s",&ans);
}while (ans=='y'|| ans=='Y');


printf ("\n Value of list from start to end \n");
traversal_S_to_E ();
printf ("\n Value of list from end to start \n");
traversal_E_to_S ();

}
```

Output:

```
Enter value to be inserted  55

Cont...(y/n) y

Enter value to be inserted  33

Cont...(y/n) y

Enter value to be inserted  56

Cont...(y/n) n

 Value of list from start to end
55
 33
 56

 Value of list from end to start
56
 33
 55
```

Try yourself:-

(1) Write a program to implement circular Linked List.

(2) Write a program to merge two linked lists.

(3) Write a program to sort a linked list.

Algorithm: For inorder traversal

A binary tree t is in memory. This algorithm does an inorder traversal of t. An array stack is used to temporarily hold the address of nodes.

INORDER_TRAVERSAL (INFO, LEFT, RIGHT, ROOT)

1. [PUSH NULL TO STACK AND INITIALIZE PTR]

SET TOP: =1

NOTES

```
SET STACK [TOP]:=NULL
SET PTR: =ROOT

2.  REPEAT WHILE PTR ‘“ NULL
SET TOP: = TOP +1
SET STACK [TOP]:= PTR
SET PTR: = LEFT [PTR]
[END OF LOOP]
3.  SET PTR:= STACK[TOP]
SET TOP: = TOP -1
4.  REPEAT STEPS 5 TO 7 WHILE PTR ‘“ NULL
5.  APPLY PROCESS TO INFO[PTR]
6.  [RIGHT CHILD?]
IF RIGHT [PTR] ‘“ NULL THEN
SET PTR: = RIGHT [PTR]
GOTO STEP 2
[END OF IF]
7.  SET PTR:= STACK[TOP]
SET TOP: = TOP -1
[END OF STEP 4 LOOP]
8.  END
```

Algorithm: For pre-order traversal

A binary tree t is in memory. This algorithm does preorder traversal of t. An array stack is used to temporarily old the address of nodes.

```
PREORDER _TRAVERSAL (INFO, LEFT, RIGHT, ROOT)
1.  [INITIALLY PUSH NULL TO STACK AND INITIALIZE PTR]
SET TOP: =1
SET STACK [TOP]:=NULL
SET PTR: =ROOT
2.  REPEAT STEPS 3 TO 5 WHILE PTR ‘“ NULL
3.  APPLY PROCESS TO INFO[PTR]
4.  [RIGHT CHILD?]
IF RIGHT [PTR] ‘“ NULL THEN
SET PTR: = RIGHT [PTR]
SET TOP: = TOP +1
SET STACK [TOP]:= RIGHT [PTR]
 [END OF IF]

5.  [LEFT CHILD?]
IF LEFT [PTR] ‘“ NULL THEN
SET PTR: = LEFT [PTR]
```

ELSE
SET PTR: = STACK [TOP]
SET TOP: = TOP -1
[END OF IF]
[END OF LOOP]
6. END

Algorithm: For post-order traversal

A binary tree t is in memory. This algorithm does postorder traversal of t. An array stack is used to temporarily old the address of nodes.

POSTORDER _TRAVERSAL (INFO, LEFT, RIGHT, ROOT)
1. [INITIALLY PUSH NULL TO STACK AND INITIALIZE PTR]
SET TOP: =1
SET STACK [1]:=NULL
SET PTR: =ROOT
2. [PUSH LEFT-MOST PATH ONTO STACK]
REPEAT STEPS 3 TO 5 WHILE PTR ‘“ NULL
3. SET TOP:=TOP+1
SET STACK [TOP]:=PTR

[PUSHES PTR ON STACK]
4. IF RIGHT[PTR] ‘“ NULL THEN [PUSH ON STACK]
SET TOP: =TOP+1
SET STACK [TOP]:= -RIGHT [PTR]
[END OF IF STRUCTURE]
5. SET PTR:= LEFT[PTR] [UPDATE POINTER PTR]
[END OF STEP 2 LOOP]

6. [POP NODE FROM STACK]
SET PTR: = STACK [TOP]
SET TOP: =TOP-1
7. REPEAT WHILE PTR>0
a. APPLY PROCESS TO INFO[PTR]
b. [POP NODE FROM STACK]
SET PTR: = STACK [TOP]
SET TOP: =TOP-1
[END OF LOOP]
8. IF PTR<0 THEN
a) SET PTR:= -PRT
b) GOTO STEP 2
[END OF IF STRUCTURE]
9. END

36. Write a program for binary tree insertion and inorder traversal.

NOTES

```
//C program for inorder traversal of binary tree


#include <stdio.h>
#include <stdlib.h>


//structure for NODE
struct node
{
 int data;
 struct node *left;
 struct node *right;
}*root;


 // Function to inserting value into Tree
void insert (struct node *tree, struct node *newnode)
{
 if (root == NULL)
 {
 root = (struct node*)malloc(sizeof(struct node));


 root->data = newnode->data;
 root->left = NULL;
 root->right = NULL;
 printf ("Root Node is Added \n");
 return;
 }
 if (tree->data == newnode->data)
 {
 printf ("Element already in the tree \n");
 return;
 }
 if (tree->data > newnode->data)
 {
 if (tree->left != NULL)
 {
```

NOTES

```
 insert (tree->left, newnode);
 }
 else
 {
 tree->left = newnode;
 (tree->left)->left = NULL;
 (tree->left)->right = NULL;
 printf ("Node Added To Left \n");
 return;

 }
 }
 else
 {
 if (tree->right != NULL)
 {
 insert(tree->right, newnode);
 }
 else
 {
 tree->right = newnode;
 (tree->right)->left = NULL;
 (tree->right)->right = NULL;
 printf ("Node Added To Right \n");
 return;

 }
 }
}

 // Function for inorder tree traversal
void inorder(struct node *ptr)
{
 if (root == NULL)
 {
 printf ("\n Underflow\n");
 return;

 }
 if (ptr != NULL)
```

NOTES

```
 {
 inorder (ptr->left);
 printf ("%d ", ptr->data);
 inorder (ptr->right);
 }
}

 // function to display tree structure
 void display(struct node *ptr, int level)
{
 int i;
 if (ptr != NULL)
 {
 display (ptr->right, level+1);
 printf ("\n");
 if (ptr == root)
 printf ("Root->: ");
 else
 {
 for (i = 0;i < level;i++)
 printf (" ");
 }
 printf ("%d",ptr->data);
 display (ptr->left, level+1);
}
}

int main()
{

 int ch, num;

 struct node *temp;
 char ans;
 root = NULL;
```

NOTES

```
do
{
printf ("\n1.Insert Element \n");
printf ("2.Inorder Traversal \n");
printf ("3.Display Tree Structure \n");
printf ("4. Exit \n");

printf ("Enter your choice : ");
scanf ("%d",&ch);
switch (ch)
{
case 1:

   temp = (struct node*)malloc(sizeof(struct node));
   printf ("Enter value to be inserted : ");
   scanf ("%d",&temp->data);
   insert (root, temp);
   break;
case 2:
   printf ("Inorder Traversal of BST: \n");
   inorder (root);
   break;
case 3:
   printf ("Display BST: \n");
   display (root,1);
   break;
case 4:
    exit(0);
    break;
default:
    printf ("Invalid choice \n");
}
printf ("\n\nCont...(y/n)");
scanf ("%s",&ans);
}while (ans=='y'|| ans=='Y');

}
```

Output:

NOTES

```
1.Insert Element
2.Inorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice :  1
Enter value to be inserted :  2
Root Node is Added


Cont...(y/n) y

1.Insert Element
2.Inorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice :  1
Enter value to be inserted :  33
Node Added To Right


Cont...(y/n) y

1.Insert Element
2.Inorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice :  1
Enter value to be inserted :  4
Node Added To Left


Cont...(y/n) y
```

37. Write a program for binary tree insertion and preorder traversal.

```
//C program for preorder traversal of binary tree

#include <stdio.h>
#include <stdlib.h>


//structure for NODE
struct node
{
 int data;
 struct node *left;
 struct node *right;
}*root;


 // Function to inserting value into Tree
void insert(struct node *tree, struct node *newnode)
{
 if (root == NULL)
```

```
{
root = (struct node*)malloc(sizeof(struct node));


root->data = newnode->data;
root->left = NULL;
root->right = NULL;
printf("Root Node is Added \n");
return;
}
if (tree->data == newnode->data)
{
printf ("Element already in the tree \n");
return;
}
if (tree->data > newnode->data)
{
if (tree->left != NULL)
{
insert (tree->left, newnode);
}
else
{
tree->left = newnode;
(tree->left)->left = NULL;
(tree->left)->right = NULL;
printf ("Node Added To Left \n");
return;

}
}
else
{
if (tree->right != NULL)
{
insert (tree->right, newnode);
}
else
{
tree->right = newnode;
```

NOTES

NOTES

```
 (tree->right)->left = NULL;
 (tree->right)->right = NULL;
 printf ("Node Added To Right \n");
 return;

 }
 }
}

 // Function for preorder tree traversal
void preorder(struct node *ptr)
{
 if (root == NULL)
 {
 printf("\nUnderflow\n");
 }
 if (ptr != NULL)
 {
 printf ("%d ",ptr->data);
 preorder (ptr->left);
 preorder (ptr->right);
 }
}



 // function to display tree structure
 void display(struct node *ptr, int level)
{
 int i;
 if (ptr != NULL)
 {
 display (ptr->right, level+1);
 printf ("\n");
 if (ptr == root)
 printf ("Root->: ");
 else
 {
 for (i = 0;i < level;i++)
 printf (" ");
```

```
 }
 printf ("%d",ptr->data);
 display (ptr->left, level+1);
}
}

int main()
{

 int ch, num;

 struct node *temp;
 char ans;
 root = NULL;

 do
 {
 printf ("\n1.Insert Element \n");
 printf ("2.Preorder Traversal \n");
 printf ("3.Display Tree Structure \n");
 printf ("4. Exit \n");

 printf ("Enter your choice : ");
 scanf ("%d",&ch);
 switch (ch)
 {
 case 1:

    temp = (struct node*)malloc(sizeof(struct node));
    printf ("Enter value to be inserted : ");
    scanf ("%d",&temp->data);
    insert (root, temp);
    break;
 case 2:
    printf ("Preorder Traversal of BST: \n");
    preorder (root);
     break;
 case 3:
    printf ("Display BST: \n");
    display (root,1);
    break;
```

NOTES

NOTES

```
case 4:
 exit (0);
break;
default:
    printf ("Invalid choice \n");
}
printf ("\n\nCont...(y/n)");
scanf ("%s",&ans);
}while (ans=='y'|| ans=='Y');

}
```

Output:

```
1.Insert Element
2.Preorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice :  1
Enter value to be inserted :  33
Root Node is Added


Cont...(y/n) y

1.Insert Element
2.Preorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice :  1
Enter value to be inserted :  3
Node Added To Left


Cont...(y/n) y

1.Insert Element
2.Preorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice :  1
Enter value to be inserted :  77
Node Added To Right


Cont...(y/n) y
```

38. Write a program for binary tree insertion and post-order traversal.

```
//C Program for postorder traversal of binary tree

#include <stdio.h>
#include <stdlib.h>

//structure for NODE
struct node
{
```

```
 int data;
 struct node *left;
 struct node *right;
}*root;


 // Function to insert value into Tree
void insert(struct node *tree, struct node *newnode)
{
 if (root == NULL)
 {
 root = (struct node*)malloc(sizeof(struct node));


 root->data = newnode->data;
 root->left = NULL;
 root->right = NULL;
 printf ("Root Node is Added \n");
 return;
 }
 if (tree->data == newnode->data)
 {
 printf ("Element already in the tree \n");
 return;
 }
 if (tree->data > newnode->data)
 {
 if (tree->left != NULL)
 {
 insert (tree->left, newnode);
 }
 else
 {
 tree->left = newnode;
 (tree->left)->left = NULL;
 (tree->left)->right = NULL;
 printf ("Node Added To Left \n");
 return;

 }
 }
```

NOTES

NOTES

```
else
{
if (tree->right != NULL)
{
insert (tree->right, newnode);
}
else
{
tree->right = newnode;
(tree->right)->left = NULL;
(tree->right)->right = NULL;
printf ("Node Added To Right \n");
return;

}
}
}

// Function for postorder tree traversal

void postorder(struct node *ptr)
{
if (root == NULL)
{
printf ("\nUnderflow\n");
}
if (ptr != NULL)
{
postorder (ptr->left);
postorder (ptr->right);
printf ("%d ",ptr->data);

}
}

// function to display tree structure
void display(struct node *ptr, int level)
{
int i;
if (ptr != NULL)
{
```

NOTES

```
 display (ptr->right, level+1);
 printf ("\n");
 if (ptr == root)
 printf ("Root->: ");
 else
 {
 for (i = 0;i < level;i++)
 printf(" ");
 }
 printf ("%d",ptr->data);
 display (ptr->left, level+1);
}
}

int main()
{

 int ch, num;

 struct node *temp;
 char ans;
 root = NULL;

 do
 {
 printf ("\n1.Insert Element \n");
 printf ("2.Postorder Traversal \n");
 printf ("3.Display Tree Structure \n");
 printf ("4. Exit \n");

 printf ("Enter your choice : ");
 scanf ("%d",&ch);
 switch (ch)
 {
 case 1:

    temp = (struct node*)malloc(sizeof(struct node));
    printf ("Enter value to be inserted : ");
    scanf ("%d",&temp->data);
     insert (root, temp);
     break;
```

NOTES

```
case 2:
    printf ("Postorder Traversal of BST: \n");
   postorder (root);
   break;
case 3:
   printf ("Display BST: \n");
   display (root,1);
   break;
case 4:
    exit(0);
break;
default:
    printf ("Invalid choice \n");
}
printf ("\n\nCont...(y/n)");
scanf ("%s",&ans);
}while (ans=='y'|| ans=='Y');

}
```

Output:

```
1.Insert Element
2.Postorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice :  1
Enter value to be inserted :  4
Root Node is Added


Cont...(y/n) y

1.Insert Element
2.Postorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice :  1
Enter value to be inserted :  33
Node Added To Right


Cont...(y/n) y

1.Insert Element
2.Postorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice :  1
Enter value to be inserted :  2
Node Added To Left


Cont...(y/n) y

1.Insert Element
2.Postorder Traversal
3.Display Tree Structure
4. Exit
Enter your choice :  2
Postorder Traversal of BST:
2   33   4
```

Try yourself:-

(1) Write a program to check whether a tree is a binary search tree.

(2) Write a program to search an element in a tree recursively.

(3) Write a program for depth first binary tree search using recursion.

(4) Write a program to find the largest value in a tree using inorder traversal.

NOTES

Shortest Path Algorithms

1. Dijkstra's Algorithm

Dijkstra's algorithm maintains a set *S* of vertices where minimum paths have been found.

Algorithm: Dijkstra's algorithm for the single shortest path problem

Procedure: DIJKSTRA_SSSP(N, COST)

```
/*N is the number of vertices labelled {1,2,3,....N} of the weighted digraph.
COST[1:N,1:N] is the cost matrix of the graph. If there is no edge then COST
[i,j] = "*/

/* The procedure computes the cost of the shortest path from vertex 1 the
source to every other vertex of the weighted digraph*/

    T={1}; /* Initialize T to source vertex */

        for i=2 to N do
        DISTANCE[i]= COST [1,i];  /*Initialize DISTANCE vector to the cost
                                    of the end edges connecting vertex i with
                                    the source vertex
            if there is no edge then COST[1,i]= "*/

        for i=1 to N-1 do
            Choose a vertex u in V-T such that DISTANCE[u] is a minimum;
            Add u to T;
            for each vertex w in V- T do
            DISTANCE [w]= minimum (DISTANCE[w], DISTANCE[u]+
COST[u,w]);
            end
        end
    end DIJKASTRA-SSSS
```

NOTES

2. Floyd-Warshall Algorithm

The Floyd-Warshall algorithm works based on a property of *intermediate* vertices of a shortest path. An *intermediate* vertex for a path $p = <v_1, v_2, ..., v_j>$ is any vertex other than v_1 or v_j.

Algorithm

FLOYD-WARSHALL(W)

1. n = W.rows
2. $D^{(0)} = W$
3. $\Pi^{(0)} = \pi^{(0)}_{ij}$ = NIL if i=j or $w_{ij} = \infty$
   = i if i≠j and $w_{ij} < \infty$
4. for k = 1 to n
5. let $D^{(k)} = (d^{(k)}_{ij})$ be a new *n*x*n* matrix
6. for i = 1 to n
7. for j = 1 to n
8. $d^{k}_{ij} = \min(d^{(k-1)}_{ij}, d^{(k-1)}_{ik} + d^{(k-1)}_{kj})$
9. if $d^{(k-1)}_{ij}$ d" $d^{(k-1)}_{ik} + d^{(k-1)}_{kj}$
10. $\pi^{(k)}_{ij} = \pi^{(k-1)}_{ij}$
11. else
12. $\pi^{(k)}_{ij} = \pi^{(k-1)}_{kj}$
13. return $D^{(n)}$

Basically the algorithm works by repeatedly exploring paths between every pair using each vertex as an intermediate vertex.

Finding Minimum cost Spanning Trees

1. Kruskal's Algorithm:

Kruskal's algorithm is a greedy algorithm. To find out minimum spanning tree (MST) of a connected and weighted graph we use Kruskal's algorithm. This means it finds a subset of the edges to forms a tree including each vertex in such a way that the total weight of all the edges in the tree is minimal. In case a graph is not a connected graph it finds a MST connected component and that is a minimum spanning forest.

This Algorithm finds a minimum spanning tree T of a weighted graph G.

1. Order all the edges of G according to increasing weights
2. Initialize T to be a graph consisting of same nodes as G and no edges.
3. Repeat the following M-1 times, Where M is the number of nodes in G:

```
   Add to T an Edge E of G with minimum weight such
that E does not form a cycle in T.
    End of Loop
```

4. End of Loop

2. Prims Algorithm

Prim's algorithm is a very simple modification to Dijkstra's shortest path algorithm. With Prim's algorithm, you build the minimum spanning tree node by node. You are going to maintain a "current spanning tree", which will be a subset of the nodes in the graph, and the edges that compose a minimum spanning tree of those nodes.

NOTES

Procedure PRIM (G)

```
    /* G= (V, E) is a weighted, connected undirected graph and E' is the set of
edges which are  to be extracted to obtain the minimum cost spanning tree */


    E'= ²";
    Select a minimum cost edge (u,v) from E;
    V'= {u}      /*Include u in V' */
    while V'!= V do
        Let (u,v) be the lowest cost edge such that u is in V' and v is in V –V'
        Add edge (u,v) to set E';
        Add v to set V';
    End while
end PRIM
```

Graph traversing Methods

1. Breadth First Search(BFS)
2. Depth First Search(DFS)

Algorithm for BFS (Breadth *First Search*)

This Algorithm executes a breadth first search on a graph G beginning at a starting node A

1. [Initialize all nodes to the ready state]
   `Set status: =1`
2. Put the starting node A in Queue and change its status to the
   `Set status: =2`
3. Repeat steps 4 and 5 until Queue is Empty:
4. Remove the front node N of Queue process N and change the status of N to the processed
   `State.`
   `Set status: =3`

NOTES

5. Add to the rear of Queue all the neighbors of N that are in the ready state (state=1), and

```
Change their status to the waiting state
Set status: =2
[End of step 3 loop]
   End
```

Algorithm for DFS (Depth First Search)

This algorithm executes a Depth First Search on a graph G beginning A

1. [initialize all nodes to the ready state]

```
Set status: =1
```

2. Push the starting node A onto stack and change its status to the waiting state

```
Set status: =2
```

3. Repeat step 4 and 5 until stack is empty
4. Pop the top node N of stack. process N and change its status to the processed state

```
Set status: =3
```

5. Push onto stack all the neighbors of N that are still in the ready state (status=1),and change their status to the waiting state

```
Set status: =2
[End of step 3 loop]
```

6. End

Searching and Sorting Algorithms

Searching refers to the operation of finding the location of a given item in a collection of items.

Algorithm: For sequential search

INPUT : LIST OF SIZE N, TARGET VALUE T

OUTPUT : POSITION OF T IN THE LIST

1. BEGIN
2. SET FOUND: = FALSE

```
SET I: = 0
```

3. WHILE Id"N AND FOUND IS FALSE

```
IF LIST [I] = T THEN
SET FOUND: = TRUE
EXIT
ELSE
SET I: =I+1
[END OF STEP 3 LOOP]
```

4. IF FOUND = FALSE THEN

```
WRITE: T IS NOT IN LIST
ELSE
WRITE: T IS FOUND AT I LOCATION
END OF IF]
```

5. END

39. Write a program for implementing sequential search on the given array.

```
//C program for sequential search

#include <stdio.h>

//definition of sequential_Search function
void sequential_search (int a[ ] , int size , int key)
{
 int flag , i ;
 flag =0;
 for ( i=0 ; i<size ; i++)
 {
 if ( a [i] == key )
 {
 flag = 1 ;
 break;
 }
 }
 if ( flag == 1)
 printf ("value found at %d location",i+1);
 else
 printf ("value not found");
}
void main()
{
int arr[10],i,k;
printf ("Enter 10 values");
for (i=0;i<10;i++)
scanf ("%d",arr[i]);

printf (""Enter values to be searched");
scanf ("%d",k);
//call of sequential_Search function
sequential_search (arr, 10, k);
}
```

NOTES

Output:

NOTES

```
Enter 10 values
1
2
3
4
5
6
7
8
9
10
Enter  value to be searched 5
value found at 5location
```

Algorithm: For binary search

ALGORITHM: BINARY SEARCH

INPUT : SORTED LIST OF SIZE N, KEY VALUE KEY

OUTPUT : POSITION OF KEY IN THE LIST = KEY

1. BEGIN
2. [INTIALIZE]
   ```
   SET MAX: = SIZE
   SET MIN: = 1
   SET FOUND: = FALSE
   ```
3. WHILE (FOUND IS FALSE AND MAX e"MIN)
   ```
   SET MID: = (MAX + MIN)/2
   ```
4. IF KEY = LIST [MID] THEN
   ```
   SET I : = MID
   SET FOUND: = TRUE
   EXIT
  ELSE IF KEY < LIST [MID] THEN
   SET MAX: = MID -1
  ELSE
   SET MIN: = MIN +1
  [END OF IF]
  [END OF LOOP]
   ```
5. IF FOUND = FALSE THEN
   ```
   WRITE: VALUE IS NOT IN LIST
  ELSE
   WRITE VALUE FOUND AT MID LOCATION
   ```
6. END

40. Write a C program for binary search.

NOTES

```
//C program for binary Search

#include <stdio.h>
// Binary Search Function
void binary_search (int a[ ] , int size , int key)
{
int low ,high ,mid ,flag ;
flag= 0;
low = 0;
high = size -1;
while (low <= high && flag ==0)
{
mid =(low +high)/2;
if ( key == a [mid])
{
flag=1;
break;
}
else if (key < a[mid ] )
{
high = mid -1;
}
else
{
low = mid +1;
}
}
if ( flag ==1)
{
printf ("value found at %d location ", mid +1);
}
else
printf ("value not found ");
}


void main()
{
int arr[10],i,k;
printf ("Enter 10 values\n";
```

NOTES

```
for(i=0;i<10;i++)
scanf("%d",arr[i]);
printf ("Enter value to be searched ");
scanf("%d",k);
//call of binary_Search function
binary_search(arr,10,k);
}
```

Output:

```
Enter 10 values
10
20
30
40
50
60
70
80
90
100
Enter  value to be searched 60
value found at  location6
```

Sorting refers to the operation of arranging data in some given order such as increasing or decreasing with numerical data and alphabetically with character data.

Selection Sort

Selection sort algorithm starts by comparing first two elements of an array and swapping if necessary.

This algorithm is not suitable for large data sets as its average and worst case complexities are of $O(n^2)$, where **n** is the number of items.

41. Write a program to sort an array using selection sort.

```
//C program for selection sort
#include <stdio.h>
void selection_sort (int a[ ], int size )
{
 int temp ,i,j, min;

 for(int i = 0; i < size-1 ; i++) {

 min = i ; //considering element i as minimum

 for(int j = i+1; j < size ; j++ )
 {
 if(a[ j ] < a[ min ])
```

```
 {
 min = j ;
 }
 }
 temp= a[ min ];
 a[ min ]=a[ i ] ;
 a[ i]=temp;

 }
}
//main function
void main()
{
int arr[10],i;
cout<<"Enter 10 values\n";
for(i=0;i<10;i++)
cin>>arr[i];

//call of selection sort function
selection_sort(arr,10);
cout<<" \n Sorted Values \n";
for(i=0;i<10;i++)
cout<<endl<<arr[i];
}
```

Output:

```
Enter 10 values
99
8
67
5
6
34
78
1
2
43

 Sorted Values

1
2
5
6
8
34
43
67
78
99
```

NOTES

NOTES

42. Write a program to sort an array using bubble sort.

```
//C program for bubble sort
#include <stdio.h>
void bubble_sort (int a[ ], int size )
{
 int temp ,i,j;
 for(i=0; i<size; i++)
 {
 for(j=0; j<size-1; j++)
 {

 if(a[j]>a[j+1])
 {
 temp=a[j];
 a[j]=a[j+1];
 a[j+1]=temp;
 } }
 }
}
/main function

void main()
{
int arr[10],i;
printf("Enter 10 values\n");
for(i=0;i<5;i++)
scanf("%d",arr[i]);

//call of bubble sort function
bubble_sort(arr,10);

printf("\n Sorted Values \n");
for(i=0;i<5;i++)
printf("\n %d",arr[i]);
}
```

Output:

```
Enter 10 values
65
7
4
3
44
23
4
56
9
7

 Sorted Values

3
4
4
7
7
9
23
44
56
65
```

NOTES

Insertion Sort Algorithm

ALGORITHM - INSERTION SORT

INPUT - LIST [ ] OF N ITEMS

OUTPUT - LIST [ ] OF N ITEMS IN SORTED ORDER

1. BEGIN
2. FOR I = 2 TO N DO
3. IF LIST [I] < LIST [I -1] THEN
4. SET I : = I - 1
5. SET TEMP : = LIST [I]
6. REPEAT
7. SET LIST [J + 1] = LIST [J];
8. SET I : = J - 1
9. UNTIL (J ≥ 1 AND LIST [J]> TEMP)
10. SET LIST [J+1] : =TEMP

    [END OF IF]

    [END OF STEP 2 LOOP]

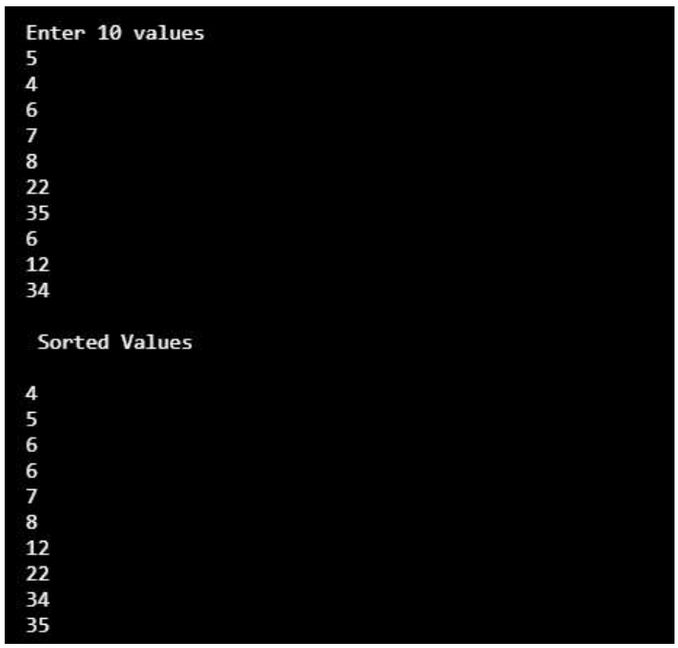
11. END

NOTES

43. Write a program to sort an array using insertion sort.

```
//C program for insertion sort Search

#include <stdio.h>

void insert_sort(int a[ ],int size)
{
int i,temp,j;


for (i = 1; i < size; i++)
 {
 temp = a[i];
 j = i-1;
while (j >= 0 && a[j] > temp)
 {
 a[j+1] = a[j];
 j—;
 }
 a[j+1] = temp;

}
}

//main function
void main()
{
int arr[10],i,k;
printf(“Enter 10 values\n”);
for(i=0;i<5;i++)
scanf(“%d”,arr[i]);


//call of Insertion Sort function
insert_sort (arr,5);

printf(“\n Sorted Values \n”);
for(i=0;i<5;i++)
printf(“\n %d”,arr[i]);

}
```

Output:



NOTES

Algorithm: For quick sort

QUICK_SORT (ARRAY, FIRST, LAST)

1. SET LOW: = FIRST
   SET HIGH: = LAST
   SET PIVOT: =ARRAY [(LOW + HIGH) /2]
2. REPEAT THROUGH STEP 7 WHILE (LOW ≤ HIGH)
3. REPEAT STEP 4 WHILE (ARRAY [LOW] <PIVOT)
4. SET LOW: = LOW+1
5. REPEAT STEP 6 WHILE (ARRAY [HIGH]>PIVOT)
6. SET HIGH: = HIGH-1
7. IF (LOW ≤ HIGH)
   ARRAY [LOW] = ARRAY [HIGH]
   SET LOW: = LOW+1
   SET HIGH: = HIGH-1
8. IF (FIORST<HIGH) THEN
   QUICK_SORT (ARRAY, FIRST, HIGH)
9. IF (LOW < LAST)
   QUICK_SORT (ARRAY, LOW, LAST)
10. END

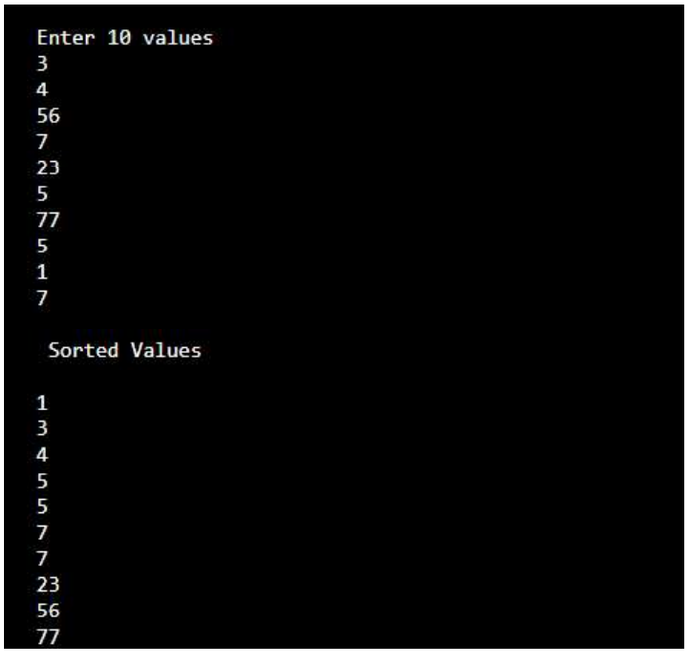
44. Write a program to sort an array using quick sort.

NOTES

```
//C program for quick sort

#include <stdio.h>

//quick sort function definition
void quick_sort (int a[ ], int first, int last)
{
int low ,high ,pivot, temp, i ;
low = first ;
high =last ;
pivot =a[(first +last)/2];

do
{
 while (a[low]<pivot)
 {
 low ++;
 }
 while (a [high]>pivot)
 {
 high—;
 }

 if (low <=high)
{
temp = a [low];
a [low]= a[high];
a [high]= temp ;
low++;
high—;
}

}while (low <=high);

if (first <high)
{
quick_sort (a, first, high);
}
if(low< last)
```

```
{
quick_sort (a, low, last);
}
}


//main function
void main()
{
int arr[10],i,k;
printf ("Enter 10 values\n");
for (i=0;i<10;i++)
scanf ("%d",arr[i]);


//call of Quick Sort function
quick_sort (arr, 0, 10);

printf ("\n Sorted Values \n");
for (i=0;i<10;i++)
printf ("\n %d",arr[i]);
}
```

Output:



Algorithm: For shell sort

ALGORITHM _ SHELL SHORT

INPUT_ LIST OF N ELEMENTS

NOTES

OUTPUT_ LIST OF N ELEMENTS IN ASSENDING ORDER

SHELL_ SORT (LIST, SIZE)

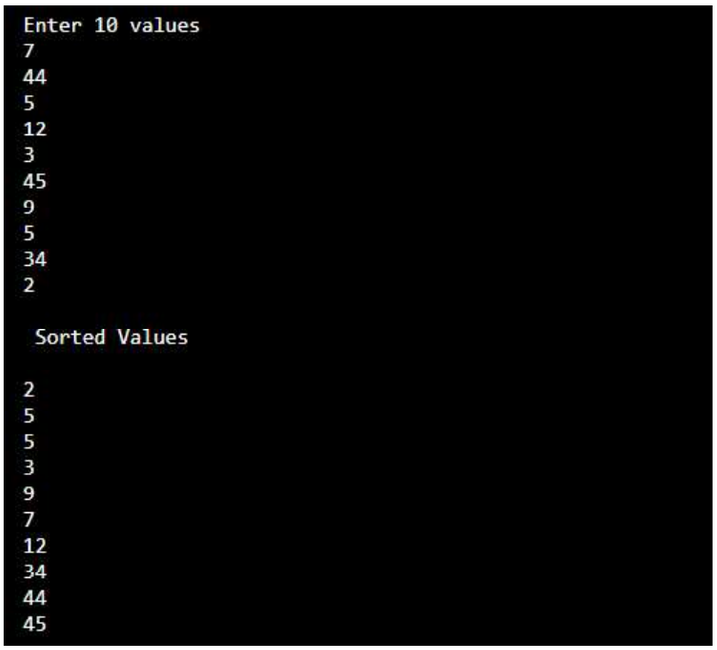
1. [INITIALIZE]
   SET GAP: = N/2
2. REPEAT THROUGH STEP 6 WHILE GAP =0
   SET SWAP: = 0
4. REPEAT THROUGH STEP 6 WHILE SWAP=1
5. REPEAT THROUGH STEP 6 FOR I=1, 3 ...I< (N-GAP)
6. IF (LIST [I] > LIST [I+ GAP]) THEN
   SET LIST [I] = LIST [I+ GAP]
   SET SWAP: = 1
   [END OF FOR LOOP]
   [END OF INNER WHILE LOOP]
   [END OF OUTER WHILE LOOP]
7. END

45. Write a program to implement shell sort.

```
//C program for shell sort
#include <stdio.h>
void shell_sort (int a[ ], int size )
{
 int temp , gap ,i ,swap ;
gap = size /2 ;
do
{
do
{
swap =0;
for ( i=0 ; i < size-gap ; i ++)
{
if(a[i] > a[ i+ gap])
{
temp = a[i] ;
a[i] = a [i+ gap];
a[ i+ gap]= temp;
```

```
swap=1;
}}
}while ( swap ==1);
gap= gap/2 ;
}while (gap >0) ;
}

//main function
void main()
{
int arr[10],i,k;
printf ("Enter 10 values\n");
for(i=0;i<10;i++)
scanf("%d",arr[i]);
//call of shell sort function
shell_sort(arr,10);

printf("\n Sorted Values \n");
for(i=0;i<10;i++)
printf("\n %d",arr[i]);
}
```

Output:



46. Write a program to implement merge sort.

NOTES

```
//C program for merge sort

#include <stdio.h>

// function to merge the two half into a sorted data.
void merge_array(int a[], int low, int high, int mid)
{
    // low to mid and mid+1 to high array are already
sorted
    int i, j, k;
    int temp_arr[high-low+1];
    i = low;
    k = 0;
    j = mid + 1;


    while (i <= mid && j <= high) // merging of two parts
                                     into temp array
    {
       if (a[i] < a[j])
       {
          temp_arr[k] = a[i];
          k++;
          i++;
       }
       else
       {
          temp_arr[k] = a[j];
          k++;
          j++;
       }
    }


    while (i <= mid) // insertion of remaining values
                              from i to mid into temp array.
    {
       temp_arr[k] = a[i];
       k++;
       i++;
```

```
    }

    while (j <= high) // insertion of remaining values
                              from j to high into temp array.
    {
       temp_arr[k] = a[j];
       k++;
       j++;
    }


    // assign sorted data stored in temp array to a array
       for (i = low; i <= high; i++)
    {
       a[i] = temp_arr[i-low];
    }
}

// A function to split array into two parts.
    void merge_sort(int a[], int low, int high)
{
    int mid;
    if (low < high)
    {
       mid=(low+high)/2;
       // split array into two parts
       merge_sort (a, low, mid);
       merge_sort (a, mid+1, high);

       // merge array to get sorted values
       merge_array (a, low, high, mid);
    }
}

 void main()
{
int arr[10],i,k;
printf ("Enter 10 values\n");
for (i=0;i<10;i++)
scanf ("%d",arr[i]);
```

NOTES

NOTES

```
//call of merge sort function
    merge_sort (arr, 0, 9);

printf ("\n Sorted Values \n");
for (i=0;i<10;i++)
printf ("\n %d",arr[i]);
}
```

Output:

```
Enter 10 values
88
6
5
7
45
3
4
6
7
1

 Sorted Values

1
3
4
5
6
6
7
7
45
88
```