DATA STRUCTURES LABORATORY
[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2017 -2018)
SEMESTER - III

Laboratory Code	**17CSL38**	**IA Marks**	**40**
Number of Lecture Hours/Week	**01I + 02P**	**Exam Marks**	**60**
Total Number of Lecture Hours	**40**	**Exam Hours**	**03**

CREDITS - 02

Descriptions (if any)

Implement all the experiments in C Language under Linux / Windows environment.

Laboratory Experiments:

1. Design, Develop and Implement a menu driven Program in C for the following **Array** operations
   a. Creating an Array of **N** Integer Elements
   b. Display of Array Elements with Suitable Headings
   c. Inserting an Element (**ELEM**) at a given valid Position (**POS**)
   d. Deleting an Element at a given valid Position(**POS**)
   e. Exit.

Support the program with functions for each of the above operations.

Design, Develop and Implement a Program in C for the following operationson **Strings**
   f. Read a main String (**STR),** a Pattern String (**PAT**) and a Replace String (**REP**)
   g. Perform Pattern Matching Operation: Find and Replace all occurrences of **PAT** in **STR** with **REP** if **PAT** exists in **STR.** Report suitable messages incase **PAT** does not exist in **STR**

Support the program with functions for each of the above operations.
Don't use Built-in functions.

2. Design, Develop and Implement a menu driven Program in C for the following operations on **STACK** of Integers (Array Implementation of Stack with maximum size **MAX**)
   a. ***Push*** an Element on to Stack
   b. ***Pop*** an Element from Stack
   c. Demonstrate how Stack can be used to check ***Palindrome***
   d. Demonstrate ***Overflow*** and ***Underflow*** situations on Stack
   e. Display the status of Stack
   f. Exit

Support the program with appropriate functions for each of the above operations

3. Design, Develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: **+, -, *, /, %(Remainder), ^(Power)** and **alphanumeric** operands**.**

4. Design, Develop and Implement a Program in C for the following Stack Applications
   a. Evaluation of **Suffix expression** with single digit operands and operators: **+,-, *, /, %, ^**
   b. Solving **Tower of Hanoi** problem with **n** disks

5. Design, Develop and Implement a menu driven Program in C for the following

operations on **Circular QUEUE** of Characters (Array Implementation of Queue with maximum size **MAX**)

a. Insert an Element on to Circular QUEUE
b. Delete an Element from Circular QUEUE
c. Demonstrate ***Overflow*** and ***Underflow*** situations on Circular QUEUE
d. Display the status of Circular QUEUE
e. Exit

Support the program with appropriate functions for each of the above operations

6. Design, Develop and Implement a menu driven Program in C for the following operations on **Singly Linked List (SLL)** of Student Data with the fields: ***USN, Name, Branch,***
Sem, PhNo

a. Create a SLL of N Students Data by using front insertion.
b. Display the status of SLL and count the number of nodes in it
c. Perform Insertion / Deletion at End of SLL
d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)
e. Exit

9. Design, Develop and Implement a menu driven Program in C for the following operations on **Doubly Linked List (DLL)** of Employee Data with the fields: ***SSN, Name, Dept,***
Designation, Sal, PhNo

a. Create a **DLL** of **N** Employees Data by using ***end insertion***.
b. Display the status of **DLL** and count the number of nodes in it
c. Perform Insertion and Deletion at End of **DLL**
d. Perform Insertion and Deletion at Front of **DLL**
e. Demonstrate how this **DLL** can be used as **Double Ended Queue**
f. Exit

10. Design, Develop and Implement a Program in C for the following operationson **Singly Circular Linked List (SCLL)** with header nodes

a. Represent and Evaluate a Polynomial $\mathbf{P(x,y,z) = 6x^2y^2z-4yz^5+3x^3yz+2xy^5z-2xyz^3}$
b. Find the sum of two polynomials **POLY1(x,y,z)** and **POLY2(x,y,z)** and store the result in **POLYSUM(x,y,z)**

Support the program with appropriate functions for each of the above operations

10. Design, Develop and Implement a menu driven Program in C for the following operations on **Binary Search Tree (BST)** of Integers

a. Create a BST of **N** Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2
b. Traverse the BST in Inorder, Preorder and Post Order
c. Search the BST for a given element (**KEY**) and report the appropriate message
e. Exit

11. Design, Develop and Implement a Program in C for the following operations on **Graph(G)** of Cities

a. Create a Graph of **N** cities using Adjacency Matrix.
b. Print all the nodes **reachable** from a given starting node in a digraph using DFS/**BFS** method

12. Given a File of **N** employee records with a set **K** of Keys(4-digit) which uniquely determine the records in file **F**. Assume that file **F** is maintained in memory by a Hash Table(HT) of **m** memory locations with **L** as the set of memory addresses (2-digit) of locations in HT. Let the keys in **K** and addresses in **L** are Integers. Design and develop

a Program in C that uses Hash function **H: K→L** as H(**K**)=**K** mod **m (remainder** method), and implement hashing technique to map a given key **K** to the address space **L.** Resolve the collision (if any) using **linear probing**.

Course outcomes:

On the completion of this laboratory course, the students will be able to:

- Analyze and Compare various linear and non-linear data structures
- Demonstrate the working nature of different types of data structures and their applications
- Develop, analyze and evaluate the searching and sorting algorithms
- Choose the appropriate data structure for solving real world problems

Conduction of Practical Examination:

1. All laboratory experiments (**TWELVE** nos) are to be included for practical examination.
2. Students are allowed to pick one experiment from the lot.
3. Strictly follow the instructions as printed on the cover page of answer script
4. Marks distribution: Procedure + Conduction + Viva:**15 + 70 +15 (100)**
5. **Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.**

Array operations using C

Exp1:

AIM:

Design, Develop and Implement a menu driven Program in C for the following Array operations

a. Creating an Array of N Integer Elements

b. Display of Array Elements with Suitable Headings

c. Inserting an Element (ELEM) at a given valid Position (POS)

d. Deleting an Element at a given valid Position(POS)

e. Exit.

Support the program with functions for each of the above operations

ALGORITHM:

```
Step 1: Start.
Step 2: Read number of elements.(n)
Step 3: Read Array of N integer elements
Step 4: Display array contents and checking array empty condition
Step 5: Read valid position, to insert element into that position
Step 6: Delete an element at given valid position from an array.
Step 7: Stop
```

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
int a[20];
int n,val,i,pos;

/*Function Prototype*/
#include<stdio.h>
#include<stdlib.h>
int a[20];
int n,val,i,pos;

void create();
void display();
void insert();
void delete();

int main()
{
        int choice;
        while(choice)
        {
        printf("\n\n--------MENU        \n");
        printf("1.CREATE\n");
        printf("2.DISPLAY\n");
        printf("3.INSERT\n");
        printf("4.DELETE\n");
        printf("5.EXIT\n");
        printf(" ");
        printf("\nENTER YOUR CHOICE:\t");
        scanf("%d",&choice);

switch(choice)
{
        case 1: create();
                break;
        case 2: display();
                break;
        case 3:insert();
                 break;
        case 4:delete();
                break;
        case 5:exit(0);
                 break;
        default:
                printf("\nInvalid choice:\n");
                 break;
}

}
        return 0;
}
void create()
{
        printf("\nEnter the size of the array elements:\t");
        scanf("%d",&n);
        printf("\nEnter the elements for the array:\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
}
void display()
{
        int i;
        printf("\nThe array elements are:\n");
        for(i=0;i<n;i++)
        {
        printf("%d\t",a[i]);
        }
}
void insert()
{
        printf("\nEnter the position for the new element:\t");
        scanf("%d",&pos);
        printf("\nEnter the element to be inserted :\t");
        scanf("%d",&val);
        for(i=n-1;i>=pos;i--)
        {
        a[i+1]=a[i];
        }
        a[pos]=val; n=n+1;
}
```

```
void delete()
{
        printf("\nEnter the position of the element to be deleted:\t");
        scanf("%d",&pos);
        val=a[pos];
        for(i=pos;i<n-1;i++)
        {
                a[i]=a[i+1];
        }
        n=n-1;
        printf("\nThe deleted element is =%d",val);
}
```

Sample Input and Output:

```
inux:~/dslab # gedit array.c linux:~/dslab # cc array.c linux:~/dslab # ./a.out

..........MENU................
1.        CREATE
2.        DISPLAY
3.        INSERT
4.        DELETE
5.        EXIT
--------------------------
ENTER YOUR CHOICE: 1
Enter the size of the array elements: 3 Enter the elements for the array:
 10 25 30

ENTER YOUR CHOICE: 2
The array elements are:
10 25 30

ENTER YOUR CHOICE: 3
Enter the position for the new element: 1 Enter the element to be inserted : 20

ENTER YOUR CHOICE: 2
The array elements are:
10 20 25 30

ENTER YOUR CHOICE: 4
Enter the position of the element to be deleted: 3 The deleted element is =30

enter your choice: 5
```

Pattern Matching

Exp 2:

AIM:

Design, Develop and Implement a Program in C for the following operations on Strings
Read a main String (STR), a Pattern String (PAT) and a Replace String (REP) b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR.
Support the program with functions for each of the above operations. Don't use Built-in functions.

ALGORITHM

Step 1: Start.
Step 2: Read main string STR, pattern string PAT and replace string REP.
Step 3: compare pattern string in main string,
Step 4: if PAT is found then replace all occurrences of PAT in main string STR with REP string.
Step 5: if PAT is not found give a suitable error message.
Step 6: Stop.

PROGRAM

```
#include<stdio.h>
void main()
{
        char STR[100],PAT[100],REP[100],ans[100];
        int i,j,c,m,k,flag=0;

        printf("\nEnter the MAIN string: \n");
        gets(STR);

        printf("\nEnter a PATTERN string: \n");
        gets(PAT);
        printf("\nEnter a REPLACE string:\n");
        gets(REP);
        i = m = c = j = 0;
        while ( STR[c] != '\0')
        {
                if ( STR[m] == PAT[i] )
                {
                        i++;
                        m++;
                        flag=1;
                        if ( PAT[i] == '\0')
                        {
                                for(k=0; REP[k] != '\0';k++,j++)
                                        ans[j] = REP[k];
                                        i=0;
                                        c=m;
                        }
```

```
                    }

                    else
                              {
                              ans[j] = STR[c]; j++;
                              c++;
                              m = c;
                              i=0;
                              }
}
          if(flag==0)
          {
                    printf("Pattern doesn't found!!!");
          }
          else
          {
                    ans[j] = '\0';
                    printf("\nThe RESULTANT string is:%s\n" ,ans);
          }
}
```

<u>SAMPLE INPUT AND OUTPUT:</u>

```
linux:~/dslab # gedit string.c linux:~/dslab # cc string.c linux:~/dslab # ./a.out

Enter the MAIN string:

good morning


Enter a PATTERN string: morning


Enter a REPLACE string:
evening
The RESULTANT string is: good evening


linux:~/dslab # ./a.out
Enter the MAIN string: hi vcet


Enter a PATTERN string: bye
Enter a REPLACE string: hello
Pattern doesn't found!!
```

STACK OPERATIONS

Exp 3

AIM:

Design, Develop and Implement a menu driven Program in C for the followingoperations on
STACK of Integers (Array Implementation of Stack with maximum size MAX)
a. *Push* an Element on to Stack
b. *Pop* an Element from Stack
c.* Demonstrate how Stack can be used to check *Palindrome
d. Demonstrate *Overflow* and *Underflow* situations on Stack
e. Display the status of Stack
f. Exit
Support the program with appropriate functions for each of the above operations

ALGORITHM:

```
PUSH(item)
Step 1: Read an element to be pushed on to stack item

Step 2: check overflow condition of stack before inserting element into stack Top=max-1

Step 3: update the top pointer and insert an element into stack
Top=top+1
S[top] <-item


POP(item)

Step1: check underflow condition of stack before deleting element from stack top=-1

Step2:Display deleted element pointed by top
Deleted element<- s[top]

Step3: Decrement top pointer by 1
top<-top-1

Palindrome()
Step1: two pointer is required ,one is pointed to top of stack another is bottom of stack

Step2: compare top and bottom elements of stack if it is equal update top and bottom pointer by1

Step 3: if all elements are equal, then stack content is palindrome.
```

PROGRAM

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#define max_size 5
int stack[max_size],top=-1,flag=1;
int i,temp,item,rev[max_size],num[max_size];
void push();
void pop();
void display();
void pali();
int main()
{
        int choice;
        printf("\n\n -------STACK OPERATIONS\n");
        printf("1.Push\n");
        printf("2.Pop\n");
        printf("3.Palindrome\n");
        printf("4.Display\n");
        printf("5.Exit\n");
        printf(" ");
        while(1)
        {
                printf("\nEnter your choice:\t");
                scanf("%d",&choice);

        switch(choice)
        {
                case 1:  push();
                        break;

                case 2: pop();
                        if(flag)
                        printf("\nThe poped element: %d\t",item);
                        temp=top;
                                break;
                case 3: pali();
                        top=temp;
                                break;
                case 4: display();
                        break;

                case 5:  exit(0);
                         break;
                default: printf("\nInvalid choice:\n");
                        break;
}
}
}
void push()
{
if(top==(max_size-1))
{
printf("\nStack Overflow:");
}
else
{
printf("Enter the element to be inserted:\t");
scanf("%d",&item);
top=top+1;
stack[top]=item;
}
temp=top;
}
```

```
void pop()
{
        if(top==-1)
        {
                printf("Stack Underflow:");
                flag=0;
        }

        else
        {
                item=stack[top]; top=top-1;
        }
}

void pali()
{
        i=0;
        if(top==-1)
        {
                printf("Push some elements into the stack first\n");
        }
        else
        {
        while(top!=-1)
        {
                rev[top]=stack[top]; pop();
        }
        top=temp;
        for(i=0;i<=temp;i++)
        {
                if(stack[top--]==rev[i])
                {
                        if(i==temp)
                        {
                        printf("Palindrome\n"); return;
                        }
                }
        }
        printf("Not Palindrome\n");
}
}
void display()
{
        int i; top=temp;
        if(top==-1)
        {
        printf("\nStack is Empty:");
        }
        else
        {
        printf("\nThe stack elements are:\n" );
        for(i=top;i>=0;i--)
        {
                printf("%d\n",stack[i]);
        }
}
}
```

SAMPLE INPUT AND OUTPUT

```
linux:~/dslab # gedit stack.c linux:~/dslab # cc stack.c linux:~/dslab # ./a.out


--------STACK OPERATIONS-----------
1.Push 2.Pop
3.Palindrome 4.Display 5.Exit
-------------------------
Enter your choice: 1
Enter the element to be inserted: 1

Enter your choice: 1
Enter the element to be inserted: 2

Enter your choice: 1
Enter the element to be inserted: 1

Enter your choice: 1
Enter the element to be inserted: 5

Enter your choice: 2 The poped element: 5


Enter your choice: 4 The stack elements are: 1
2
1
Enter your choice: 3 Numbers= 1
Numbers= 2
Numbers= 1 reverse operation : reverse array : 1
2
1
check for palindrome : It is palindrome number

Enter your choice: 5
```

Infix to postfix conversion

Exp No 4

AIM:

Design, Develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, %(Remainder), ^(Power) and alphanumeric operands.

ALGORITHM

Step 1: Read the infix expression as a string.

Step 2: Scan the expression character by character till the end. Repeat the following operations

If it is an operand add it to the postfix expression.

If it is a left parenthesis push it onto the stack.

If it is a right parentheses pop out elements from the stack and assign it to the postfix string. Pop out the left parentheses but dont assign to postfix.

Step 3: If it is an operator compare its precedence with that of the element at the top of stack.

1.If it is greater push it onto the stack.

2. Else pop and assign elements in the stack to the postfix expression until you find one such element.

Step 4: If you have reached the end of the expression, pop out any leftover elements in the stack till it becomes empty.

Step 5: Append a null terminator at the end display the result

PROGRAM:

```
#define SIZE 50 /* Size of Stack */
#include <ctype.h>
#include <stdio.h>
char s[SIZE];
int top = -1; /* Global declarations */
push(char elem) /* Function for PUSH operation */
{
        s[++top] = elem;
}

char pop() /* Function for POP operation */
{
        return (s[top--]);
```

```
}

int pr(char elem) /* Function for precedence */
{
    switch (elem)
    {
        case '#': return 0;
        case '(': return 1;
        case '+':
        case '-': return 2;
        case '*': case '/':
        case '%': return 3;
        case '^': return 4;
    }
}

void main() /* Main Program */
{
    char infx[50], pofx[50], ch, elem;
    int i = 0, k = 0;
    printf("\n\nRead the Infix Expression ? ");
    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);
            }
        }

        while (s[top] != '#') /* Pop from stack till empty */
                pofx[k++] = pop();
                pofx[k] = '\0'; /* Make pofx as valid string */
                printf("\n\nGiven Infix Expn: %s Postfix Expn: %s\n", infx, pofx);
}
```

SAMPLE INPUT AND OUTPUT

```
linux:~/dslab # gedit intopost.c linux:~/dslab # cc intopost.c linux:~/dslab # ./a.out
Read the Infix Expression ? (a+b)*c/d^5%1
Given Infix Expn: (a+b)*c/d^5%1

Postfix Expn: ab+c*d5^/1%
```

Evaluation of suffix expression and tower of Hanoi problem

Exp No 5

AIM:

Design, Develop and Implement a Program in C for the following Stack Applications a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^ b. Solving Tower of Hanoi problem with n disks

ALGORITHM

Step 1: Read the suffix/postfix expression

Step 2: Scan the postfix expression from left to right character by character

Step 3: if scanned symbol is operand push data into stack.

If scanned symbol is operator pop two elements from stack Evaluate result and result is pushed onto stack

Step 4: Repeat step 2-3 until all symbols are scanned completely

PROGRAM

```
#include <stdio.h>
//#include <conio.h>
#include <math.h>
#define MAX 20

struct stack
{
        int top;
        float str[MAX];
}s;

char postfix[MAX];//postfix
void push(float);
float pop();
int isoperand(char);
float operate(float,float,char);

int main()
{
        int i=0;
        printf("Enter Expression:");
        scanf("%s",postfix);

        float ans,op1,op2;
        while(postfix[i]!='\0')
        {
        if(isoperand(postfix[i]))
                push(postfix[i]-48);
        else
        {
                op1=pop();
        op2=pop();
        ans=operate(op1,op2,postfix[i]);
        push(ans);
        printf("%f %c %f = %f\n",op2,postfix[i],op1,ans);
    }
    i++;
}
    printf("%f",s.str[s.top]);
    getchar();
}

int isoperand(char x)
{
    if(x>='0' && x<='9')
    return 1;
    else
    return 0;
}
void push(float x)
{
    if(s.top==MAX-1)
        printf("Stack is full\nStack overflow\n");
    else
    {
        s.top++;
        s.str[s.top]=x;
    }
}

float pop()
{
    if(s.top==-1)
    {
        printf("Stack is empty\nSTACK UNDERFLOW\n");
        getchar();
    }
    else
    {
        s.top--;
        return s.str[s.top+1];
    }
}

float operate(float op1,float op2,char a)
{
switch(a)
{
    case '+' : return op2+op1;
    case '-' : return op2-op1;
    case '*' : return op2*op1;
    case '/' : return op2/op1;
    case '^' : return pow(op2,op1);
}
}


// 5a. Towers of Hanoi
```

ALGORITHM:

Step 1: Read No of disks called n from keyboard.
Step 2: Check if nis not zero or a negative no. if yes display suitable message else go to step3.
Step 3: Call tower of Hanoi function with n as parameter,
Step 4: Stop

TOWERS OF HANOI FUNCTION TO MOVE DISKS FROM A TO C USING B()

Step 1: If n is equal to 1 then move the single disk from A to C and stop
Step 2: Move the top n-1

Step 1 disks from A to B using c as auxiliary.

Step 3: Move the remaining disk from A to C.
Step 4: Move the n-1 disks from B to C using as auxillary.

PROGRAM

```
#include<stdio.h>
void towers(int, char, char, char);
int main()
{
        int num;
        printf("Enter the number of disks : ");
        scanf("%d", &num);

        printf("The sequence of moves involved in the Tower of Hanoi are :\n");
        towers(num, 'A', 'C', 'B');
        return 0;
}

void towers(int num, char frompeg, char topeg, char auxpeg)
{
        if (num == 1)
        {
                printf("\n Move disk 1 from peg %c to peg %c", frompeg,topeg);
                return;
        }
towers(num - 1, frompeg, auxpeg, topeg);

printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);
towers(num - 1, auxpeg, topeg, frompeg);
}
```

SAMPLE INPUT AND OUTPUT

```
linux:~/dslab #gedit posteval.c linux:~/dslab #gcc posteval.c -lm linux:~/dslab # ./a.out

Insert a postfix notation :: 22^32*+
Result :: 10
```

```
linux:~/dslab #gedit tower.c linux:~/dslab #gcc tower.clinux:~/dslab # ./a.out

 Enter the number of disks : 3
The sequence of moves involved in the Tower of Hanoi are :
 Move disk 1 from peg A to peg C


Move disk 2 from peg A to peg B
Move disk 1 from peg C to peg B
Move disk 3 from peg A to peg C
Move disk 1 from peg B to peg A
Move disk 2 from peg B to peg C
Move disk 1 from peg A to peg C
```

Circular queue

Exp 6

AIM:

Design, Develop and Implement a menu driven Program in C for the following operations

on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

a. Insert an Element on to Circular QUEUE

b. Delete an Element from Circular QUEUE

c. Demonstrate*Overflow*and*Underflow*situations on Circular QUEUE d. Display the status of Circular QUEUE

e. Exit

Support the program with appropriate functions for each of the above operations

ALGORITHM

Step1: Initialize front and rear pointer and also count

front->0,count<-0,rear<- -1

Step2: Insert an element into queue before check overflow condition Count=max

Insert an element rear<-(rear+1) %max

q[rear]<-item and count=count+1

Step3: Delete an element from queue .check underflow condition

Count=0 underflow condition. Count<-count-1

Item<-q[front]Deleted element

Step4: Display contents of queue. Number of elements represents count .

Check empty queue condition before displaying an element

PROGRAM

```
#include <stdio.h>
//#include <conio.h>
#include <stdlib.h>
#define SIZE 5

int CQ[SIZE];
int front=-1;
int rear=-1, ch;
int IsCQ_Full();
int IsCQ_Empty();

void CQ_Insert(int );
void CQ_Delet();
void CQ_Display();
void main()
{
    printf("1.Insert\n2.Delete\n3.Display\n4.Exit\n");
    while(1)
    {
        int ele;
        printf("Enter your choice\n");
        scanf("%d",&ch);
    switch(ch)
    {
    case 1: if(IsCQ_Full())
        printf("Circular Queu Overflow\n");
       else
        {
        printf("Enter the element to be inserted\n");
        scanf("%d",&ele);
        CQ_Insert(ele);
        }
       break;
    case 2: if(IsCQ_Empty())
        printf("Circular Queue Underflow\n");
       else
        CQ_Delet();
       break;
    case 3: if(IsCQ_Empty())
        printf("Circular Queue Underflow\n");
       else
        CQ_Display();
       break;
    case 4: exit(0);
    }
    }
}

void CQ_Insert(int item)
{
    if(front==-1)
        front++;
        rear = (rear+1)%SIZE;
        CQ[rear] =item;
}

void CQ_Delet()
{
    int item;
    item=CQ[front];
    printf("Deleted element is: %d",item);
    front = (front+1)%SIZE;
}

void CQ_Display()
{
    int i;
    if(front==-1)
        printf("Circular Queue is Empty\n");
    else
    {
        printf("Elements of the circular queue are..\n");
        for(i=front;i!=rear;i=(i+1)%SIZE)
        {
            printf("%d\t",CQ[i]);
        }
        printf("%d\n",CQ[i]);
        }
}

int IsCQ_Full()
{
        if(front ==(rear+1)%SIZE)
                return 1;
        return 0;
}

int IsCQ_Empty()
{
        if(front == -1)
                return 1;
        else
                if(front == rear)
                {
                        printf("Deleted element is: %d",CQ[front]);
                        front=-1;
                        return 1;
                }
        return 0;
}
```

SAMPLE INPUT AND OUTPUT

```
linux:~/dslab #gedit cirQ.c linux:~/dslab #gcc cirQ.c linux:~/dslab # ./a.out

Circular Queue operations 1.insert
2.delete 3.display 4.exit
Enter your choice:1
Enter element to be insert:10

Enter your choice:1
Enter element to be insert:20

Enter your choice:1

Enter element to be insert:30

Enter your choice:3 10 20 30
rear is at 30 front is at 10
Enter your choice:2 Deleted element is:10

Enter your choice:3 20 30
rear is at 30 front is at 20
Enter your choice:4
```

Singly Linked List

Exp No 7

AIM

Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: *USN, Name, Branch,*
Sem, PhNo

a. Create a SLL of N Students Data by using*front insertion*.
b. Display the status of SLL and count the number of nodes in it c. Perform Insertion and Deletion at End of SLL d. Perform Insertion and Deletion at Front of SLL
e. Demonstrate how this SLL can be used as STACK and QUEUE f. Exit

ALGORITHM

```
Step 1: declare structure of node create empty list
head->null

Step2: Insert at front end
head<-null
return temp
if list is empty
temp->link=head
return head

Step 3:Insert at rear end
head=null
return temp
if list is empty
cur->head
while(cur!=null)
cur=cur->link
cur->link=temp;
return head

Step 4: Delete at front end
head->link=null;
return null
if list has only one node
cur=head
head=head->link
free(cur)

Step 5:Delete at Rear end
head->link=null
```

```
return null

if only one node
cur<-head
while(cur!=null)
prev<-cur, cur=cur<-link;
free(cur);
```

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int count=0;
struct stud
{
        long long int ph;
        int sem;
        char name[15],usn[15],brnch[8];
        struct stud *next;
}*head=NULL,*tail=NULL,*temp=NULL,*temp1;

void create(long long int n,int s,char na[20],char u[15],char b[5])
{
        if(head==NULL)
        {
                head=(struct stud*)malloc(1*sizeof(struct stud));
                head->ph=n;
                head->sem=s;
                strcpy(head->name,na);
                strcpy(head->usn,u);
                strcpy(head->brnch,b);
                head->next=NULL;
                tail=head;
                count++;
        }
        else
        {
                temp=(struct stud*)malloc(1*sizeof(struct stud));
                temp->ph=n;
                temp->sem=s;
                strcpy(temp->name,na);
                strcpy(temp->usn,u);
                strcpy(temp->brnch,b);
                temp->next=NULL;
                tail->next=temp;
                tail=temp;
                count++;
        }
}

void display()
{
        temp1=head;

        if(temp1==NULL)
        {
                printf("\nlist is empty\n");
        }
        else
        {
                printf("student details are as follows:\n");
                while(temp1!=NULL)
                {
                        printf(" \n");
                        printf("NAME:%s\nUSN:%s\nBRANCH:%s\nSEM:%d\nPHONE
NO.:%lld\n",temp1                                       ->name,temp1->usn,temp1->brnch,temp1-
>sem,temp1->ph);
                        printf(" \n");
                        temp1=temp1->next;
                }
                printf("no. of nodes=%d\n",count);
        }
}
void insert_head(long long int n,int s,char na[15],char u[15],char b[8])
{
        temp=(struct stud*)malloc(1*sizeof(struct stud));
        temp->ph=n;
        temp->sem=s;
        strcpy(temp->name,na);
        strcpy(temp->usn,u);
        strcpy(temp->brnch,b);
        temp->next=head;
        head=temp;
        count++;
}

void insert_tail(long long int n,int s,char na[15],char u[15],char b[8])
{
        temp=(struct stud*)malloc(1*sizeof(struct stud));
        temp->ph=n;
        temp->sem=s;
        strcpy(temp->name,na);
        strcpy(temp->usn,u);
        strcpy(temp->brnch,b);
        tail->next=temp;
        temp->next=NULL;
        tail=temp;
        count++;
}

void delete_head()
{
        temp1=head;
        if(temp1==NULL)
        {
                printf("list is empty\n");
        }
        else
        {
                head=head->next;
                printf("deleted node is:\n");
                printf(" \n");
                printf("NAME:%s\nUSN:%s\nBRANCH:%s\nSEM:%d\nPHONE NO.:%lld\n",temp1->name,
                temp1->usn,temp1->brnch,temp1->sem,temp1->ph);
                printf(" \n");
                free(temp1);
                count--;
        }
}

void delete_tail()
{
        temp1=head;
        if(temp1==NULL)
        {
                printf("list is empty\n");
        }
        while(temp1->next!=tail)
        {
                temp1=temp1->next;
        }
        printf("deleted node is:\n");
        printf(" \n");
        printf("NAME:%s\nUSN:%s\nBRANCH:%s\nSEM:%d\nPHONE NO.:%lld\n",tail->name,tail-
>usn,tail->brnch,tail->sem,tail->ph);
        printf("\n");
        free(tail);
        tail=temp1;
        tail->next=NULL;
        count--;
}

void main()
{
        int choice;
        long long int ph; int sem;
```

```
            char name[20],usn[15],brnch[5];
            printf("--------MENU        \n");
            printf("1.create\n2.Insert from head\n3.Insert from tail\n4.Delete from head\5.Delete fromtail\n6.display
                        \n7.exit\n");
            printf(" \n");

while(1)
{
            printf("enter your choice\n");
            scanf("%d",&choice);
switch(choice)
{
            case 1:printf("enter the name usn branch sem phno. of the student respectively\n");
                    scanf("%s%s%s%d%lld",name,usn,brnch,&sem,&ph);
                    create(ph,sem,name,usn,brnch);
                    break;

            case 2:printf("enter the name usn branch sem phno. of the student respectively\n");
                    scanf("%s%s%s%d%lld",name,usn,brnch,&sem,&ph);
                    insert_head(ph,sem,name,usn,brnch);
                    break;

            case 3:printf("enter the name usn branch sem phno. of the student respectively\n");
                    scanf("%s%s%s%d%lld",name,usn,brnch,&sem,&ph);
                    insert_tail(ph,sem,name,usn,brnch);
                    break;

            case 4:delete_head();
                    break;

            case 5:delete_tail();
                    break;

            case 6:display();
                    break;

            case 7:exit(0);
            default:printf("invalid option\n");
}
}
}
```

SAMPLE INPUT AND OUTPUT

```
linux:~/dslab #gedit slink.c linux:~/dslab #gcc slink.c linux:~/dslab # ./a.out

–..................MENU..............................
 1– create a SLL of n emp 2 - Display from beginning 3 - Insert at end
 4- delete at end 5 - Insert at beg 6 - delete at beg 7 - exit
-------------------------------------------------------

Enter choice : 1
Enter no of students : 2
Enter usn,name, branch, sem, phno of student : 007 vijay CSE 3 121 Enter usn,name, branch, sem,
phno of student : 100 yashas CSE 3 911

Enter choice : 2
Linked list elements from begining : 100 yashas CSE 3 911
007 vijay CSE 3 121 No of students = 2


Enter choice : 3
Enter usn,name, branch, sem, phno of student : 001 raj CSE 3 111

Enter choice : 2
Linked list elements from begining :
100 yashas CSE 3 911
007 vijay CSE 3 121
001 raj CSE 3 111
No of students = 3
```

```
Enter choice : 4 001 raj CSE 3 111
Enter choice : 2
Linked list elements from begining :
100 yashas CSE 3 911
007 vijay CSE 3 121 No of students = 2

Enter choice : 5
Enter usn,name, branch, sem, phno of student : 003 harsh cse 3 111

Enter choice : 2
Linked list elements from begining : 003 harsh cse 3 111
100 yashas CSE 3 911
007 vijay CSE 3 121 No of students = 3

Enter choice : 6 003 harsh cse 3 111

Enter choice : 2
Linked list elements from begining : 100 yashas CSE 3 911
007 vijay CSE 3 121 No of students = 2

Enter choice : 7
```

Doubly Linked List (DLL)

Exp No 8:

AIM

Design, Develop and Implement a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: *SSN, Name,Dept, Designation,Sal, PhNo*

a. Create a DLL of N Employees Data by using*end insertion*.

b. Display the status of DLL and count the number of nodes in it

c. Perform Insertion and Deletion at End of DLL d. Perform Insertion and Deletion at Front of DLL

e. Demonstrate how this DLL can be used as Double Ended Queue

f. Exit

ALGORITHM

Insertion at front end of list.

Step 1: Allocate memory for temp node and assign values to node

```
Step2: if list is empty, temp is attached to list directly
head=null
return temp
if list is not empty
temp->rlink=head
head->llink=temp
return head
```

```
Insertion at rear end of list.
Step1: Read node information and allocate memory for temp node
```

```
Step2: traverse the cur node upto to end of list then attach node cur to temp
cur->rlink=temp;
temp->llink=cur
```

```
Step 3:return starting address of list
return head;
```

```
Delete from front end of list.
Step 1: check if list has only one node
```

```
head=NULL;
return null;
if list is empty
head->rlink=NULL
return NULL;
if list has o0nly one node
```

```
Step 2:otherwise first node address is shifted to next node
cur=head
head=head->rlink
free(cur)
```

```
Step 3: return starting address of list
return head
```

```
Delete node from rear end
Step 1: two pointers requires one is cur and prev
```

Cur is one which points, node to be deleted.

Step 2: Traverse the cur node upto end of list before updating current pointer save the Address to prev pointer.

```
While(cur->rlink!=null)
{
prev=cur;
cur=cur->rlink;
}
prev->rlink=null;
cur->llink=null;
free(cur);
```

Step3: return starting address of the list

PROGRAM

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

struct Enode
{
        char ssn[15];
        char name[20];
        char dept[5];
        char designation[10];
        int salary;
        long long int phno;
        struct  Enode  *left;
        struct Enode *right;
}*head=NULL;

struct Enode *tail,*temp1,*temp2;
void create(char [],char [],char [],char [],int ,long long int);
void ins_beg(char [],char [],char [],char [],int ,long long int);
void ins_end(char [],char [],char [],char [],int ,long long int);
void del_beg();
void del_end();
void display();

int count=0;
void main()
{
        int choice;
        char s[15],n[20],dpt[5],des[10];
        int sal;
        long long int p;

        printf("1.Create\n2.Display\n3.Insert at beginning\n4.Insert at End\n5.Delete at beginning\n6.Delete at End\n7.Exit\n");
        while(1)
        {
                printf("\nEnter your choice\n");
                scanf("%d",&choice);
                switch(choice)
                {

                        case 1:printf("Enter the required data(Emp no,Name,Dept,Desig,sal,phone\n)");
                              scanf("%s%s%s%s%d%lld",s,n,dpt,des,&sal,&p);
                              create(s,n,dpt,des,sal,p);
                              break;

                        case 2:display();
                              break;

                        case 3:printf("Enter the required data (Emp no,Name,Dept,Desig,sal,phone\n)");
                              scanf("%s%s%s%s%d%lld",s,n,dpt,des,&sal,&p);
                              ins_beg(s,n,dpt,des,sal,p);
                              break;

                        case 4:printf("Enter the required data(Emp no,Name,Dept,Desig,sal,phone\n)");
                              scanf("%s%s%s%s%d%lld",s,n,dpt,des,&sal,&p);
                              ins_end(s,n,dpt,des,sal,p);
```

```
                                    break;

                                case 5:del_beg();
                                    break;

                                case 6:del_end();
                                    break;

                                case 7:exit(0);

                        }
            }
}

void create(char s[15],char n[20],char dpt[5],char des[10],int sal,long long int p)
{
            if(head==NULL)
            {
                        head=(struct Enode *)malloc(1*sizeof(struct Enode));
                        strcpy(head->ssn,s);
                        strcpy(head->name,n);
                        strcpy(head->dept,dpt);
                        strcpy(head->designation,des);
                        head->salary=sal;
                        head->phno=p;
                        head->left=NULL;
                        head->right=NULL;
                        tail=head;
            }
            else
            {
                        temp1=(struct Enode *)malloc(1*sizeof(struct Enode));
                        strcpy(temp1->ssn,s);
                        strcpy(temp1->name,n);
                        strcpy(temp1->dept,dpt);
                        strcpy(temp1->designation,des);
                        temp1->salary=sal;
                        temp1->phno=p;
                        tail->right=temp1;
                        temp1->right=NULL;
                        temp1->left=tail;
                        tail=temp1;
            }
}

void display()
{
            temp1=head;
            printf("Employee Details \n");
            while(temp1!=NULL)
            {
                        printf(" \n");
                        printf("%s\n%s\n%s\n%s\n%d\n%lld\n",temp1->ssn,temp1->name,temp1->dept,temp1-
>designation,temp1->salary,temp1->phno);
                        printf(" ");
                        temp1=temp1->right;

            }
}

void ins_beg(char s[15],char n[20],char dpt[5],char des[10],int sal,long long int p)
{
            temp1=(struct Enode *)malloc(1*sizeof(struct Enode));
            strcpy(temp1->ssn,s);
            strcpy(temp1->name,n);
            strcpy(temp1->dept,dpt);
            strcpy(temp1->designation,des);
            temp1->salary=sal;
            temp1->phno=p;
            temp1->right=head;
            head->left=temp1;
            head=temp1;
            temp1->left=NULL;
}

void ins_end(char s[15],char n[20],char dpt[5],char des[10],int sal,long long int p)
{
            temp1=(struct Enode *)malloc(1*sizeof(struct Enode));
            strcpy(temp1->ssn,s);
            strcpy(temp1->name,n);
            strcpy(temp1->dept,dpt);
            strcpy(temp1->designation,des);
            temp1->salary=sal;
            temp1->phno=p;
            tail->right=temp1;
            temp1->left=tail;
            temp1->right=NULL;
            tail=temp1;
}

void del_beg()
{
            temp1=head->right;
            free(head);
            head=temp1;
            head->left=NULL;
}

void del_end()
{
            temp1=tail->left;
            free(tail);
            tail=temp1;
            tail->right=NULL;
}
```

SAMPLE INPUT AND OUTPUT

```
linux:~/dslab #gedit dlink.c linux:~/dslab #gcc dlink.c linux:~/dslab # ./a.out


......................MENU..............................
1.Create 2.Display
3. Insert at beginning 4.Insert at End 5.Delete at beginning 6.Delete at End 7.Exit
------------------------------------------------
Enter choice : 1
Enter no of employees : 2
Enter ssn,name,department, designation, salary and phno of employee : 1 RAJ SALES MANAGER
15000 911
Enter ssn,name,department, designation, salary and phno of employee : 2 RAVI HR ASST 10000 123

Enter choice : 2
Linked list elements from begining :
1 RAJ SALES MANAGER 15000.000000 911
2 RAVI HR ASST 10000.000000 123
No of employees = 2 Enter choice : 3
Enter ssn,name,department, designation, salary and phno of employee : 3 RAM MARKET
MANAGER 50000 111

Enter choice : 2
Linked list elements from begining :
```

```
1 RAJ SALES MANAGER 15000.000000 911
2 RAVI HR ASST 10000.000000 123
3 RAM MARKET MANAGER 50000.000000 111
No of employees = 3 Enter choice : 4
3 RAM MARKET MANAGER 50000.000000 111
Enter choice : 2
Linked list elements from begining :
1 RAJ SALES MANAGER 15000.000000 911
2 RAVI HR ASST 10000.000000 123
No of employees = 2

Enter choice : 5
Enter ssn,name,department, designation, salary and phno of employee :
 0 ALEX EXE TRAINEE 2000 133
Enter choice : 2
Linked list elements from begining :
0        ALEX EXE TRAINEE 2000.000000 133
1        RAJ SALES MANAGER 15000.000000 911
2         RAVI HR ASST 10000.000000 123
No of employees = 3

Enter choice : 6


0        ALEX EXE TRAINEE 2000.000000 133

Enter choice : 2
Linked list elements from begining :
1        RAJ SALES MANAGER 15000.000000 911
 2 RAVI HR ASST 10000.000000 123

No of employees = 2
Enter choice : 7
Exit
```

Singly Circular Linked List (SCLL)

Exp No 9:

AIM:

Design, Develop and Implement a Program in C for the following operations on Singl Circular Linked List (SCLL) with header nodes

a. Represent and Evaluate a Polynomial P(x,y,z) = $6x^2y^2z-4yz^5+3x^3yz+2xy^5z-2xyz^3$

b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)

Support the program with appropriate functions for each of the above operations

ALGORITHM:

Evaluate a Polynomial

Step1: allocate memory for newly created node assign values to that node

Step 2: attach newly created node to list in circular fashion.

Step3: Evaluate each node information upto header node

Addition of two Polynomial

Step1: Read exponent values and co-efficicient values for each node

Step2: newly created node are attached to polynomials (p1,p2,p3)

Step3: Addition/Evaluation of list is performed

Step 4:Result is displayed

PROGRAM:

```
#include <stdio.h>
//#include <conio.h>
#include <stdlib.h>
#include <math.h>
struct node
{
        int coeff;
        int expo;
        struct node *ptr;
};

struct node *head1,*head2,*head3, *temp,*temp1,*temp2,*temp3,*list1,*list2,*list3;
struct node *dummy1,*dummy2;
void create_poly1(int , int);
void create_poly2(int , int);
void display();
void add_poly();
void eval_poly(int );
int n,ch;
int c,e,i;
void main()
{
        int x; list1=list2=NULL;
        printf("1.Create first polynomial\n2.Create Second Polynomial\n3.Display both the
polynomials\n");
        printf("4.Add Polynomials\n5.Evaluate a Polynomial\n6.Exit\n");
        while(1)
        {
                printf("Enter choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: printf("Enter the number of terms\n");
                                scanf("%d",&n);
                                printf("Enter coefficient & power of each term\n");

                                for(i=0;i<n;i++)
                                {
                                scanf("%d%d",&c,&e);
                                create_poly1(c,e);
                                 }
                        break;

                        case 2: printf("Enter the number of terms\n");
                                scanf("%d",&n);
                                printf("Enter coefficient & power of each term\n");

                                for(i=0;i<n;i++)
                               {
                                scanf("%d%d",&c,&e);
                                create_poly2(c,e);
                               }
                        break;

                        case 3:display();
                        break;

                        case 4:add_poly();
                        break;

                        case 5:printf("Enter the value for x\n");
                               scanf("%d",&x);
                               eval_poly(x);
                        break;

                        case 6:exit(0);

                }
        }
}
void create_poly1(int c, int e)
{
        dummy1=(struct node*)malloc(1*sizeof(struct node));
        dummy1->coeff=0;
        dummy1->expo=0;
        dummy1->ptr=list1;
        if(list1==NULL)
        {
                list1=(struct node*)malloc(1*sizeof(struct node));
                list1->coeff=c;
                list1->expo=e;
                list1->ptr=list1;
                head1=list1;
                head1->ptr=dummy1;
        }
        else
        {
        temp=(struct node*)malloc(1*sizeof(struct node));
        temp->coeff=c;
        temp->expo=e;
        head1->ptr=temp;
        temp->ptr=dummy1;
        head1=temp;

        }
}

void create_poly2(int c, int e)
{
        dummy2=(struct node*)malloc(1*sizeof(struct node));
        dummy2->coeff=0;
        dummy2->expo=0;
        dummy2->ptr=list2;
        if(list2==NULL)
        {
                list2=(struct node*)malloc(1*sizeof(struct node));
        list2->coeff=c;
        list2->expo=e;
        list2->ptr=list2;
        head2=list2;
        head2->ptr=dummy2;

    }
    else
    {
        temp=(struct node*)malloc(1*sizeof(struct node));
        temp->coeff=c;
        temp->expo=e;
        head2->ptr=temp;
        temp->ptr=dummy2;
        head2=temp;

    }
}
void add_poly()
{
    temp1=list1;
    temp2=list2;
    while((temp1!=dummy1)&&(temp2!=dummy2))
    {
        temp=(struct node*)malloc(1*sizeof(struct node));
        if(list3==NULL)
        {
            list3=temp;
            head3=list3;
        }
        if(temp1->expo==temp2->expo)
        {
            temp->coeff=temp1->coeff+temp2->coeff;
            temp->expo=temp1->expo;
            temp->ptr=list3;
            head3->ptr=temp;
            head3=temp;
            temp1=temp1->ptr;
            temp2=temp2->ptr;
        }
        else if(temp1->expo>temp2->expo)
        {
            temp->coeff=temp1->coeff;
            temp->expo=temp1->expo;
            temp->ptr=list3;
            head3->ptr=temp;
            head3=temp;
            temp1=temp1->ptr;
        }
        else
        {
            temp->coeff=temp2->coeff;
            temp->expo=temp2->expo;
            temp->ptr=list3;
            head3->ptr=temp;
            head3=temp;
            temp2=temp2->ptr;
        }
    }

if(temp1==dummy1)
{
    while(temp2!=dummy2)
    {
        temp=(struct node*)malloc(1*sizeof(struct node));
        temp->coeff=temp2->coeff;
        temp->expo=temp2->expo;
        temp->ptr=list3;
        head3->ptr=temp;
        head3=temp;
        temp2=temp2->ptr;
    }
}

if(temp2==dummy2)
{
    while(temp1!=dummy1)
```

```
        {
                temp=(struct node*)malloc(1*sizeof(struct node));
                temp->coeff=temp1->coeff;
                temp->expo=temp1->expo;
                temp->ptr=list3;
                head3->ptr=temp;
                head3=temp;
                temp1=temp1->ptr;
        }
}
}

void display()
{
        temp1=list1;
        temp2=list2;
        temp3=list3;
        printf("\nPOLYNOMIAL 1:");

        while(temp1!=dummy1)
        {
                printf("%dX^%d+",temp1->coeff,temp1->expo);
                temp1=temp1->ptr;

        }
        printf("\b ");
        printf("\nPOLYNOMIAL 2:");

        while(temp2!=dummy2)
        {
                printf("%dX^%d+",temp2->coeff,temp2->expo);
                temp2=temp2->ptr;

        }
        printf("\b ");
        printf("\n\nSUM OF POLYNOMIALS:\n");
        while(temp3->ptr!=list3)
        {
                printf("%dX^%d+",temp3->coeff,temp3->expo);
                temp3=temp3->ptr;

        }
        printf("%dX^%d\n",temp3->coeff,temp3->expo);
}

void eval_poly(int x)
{
        int result=0;
        temp1=list1;
        temp2=list2;
        while(temp1!=dummy1)
        {
                result+=(temp1->coeff)*pow(x,temp1->expo);
                temp1=temp1->ptr;
        }
        printf("Polynomial 1 Evaluation:%d\n",result);

        result=0;
while(temp2!=dummy2)
{
        result+=(temp2->coeff)*pow(x,temp2->expo);
        temp2=temp2->ptr;
}
printf("Polynomial 2 Evaluation:%d\n",result);
}
```

<u>SAMPLE INPUT AND OUTPUT:</u>

```
linux:~/dslab #gedit poly.c linux:~/dslab #gcc poly.c linux:~/dslab # ./a.out

........................<< MENU >>......................
Polynomial Operations : 1.Add

2.Evaluate 3.Exit
------------------------------------------------------------
Enter your choice==>1
Enter no of terms of polynomial==>3 Enter coef & expo==>
4
3
Enter coef & expo==> 2
2
Enter coef & expo==> 5
1
The polynomial is==>5x^(1) + 2x^(2) + 4x^(3) Enter no of terms of polynomial==>3
Enter coef & expo==> 4
1
Enter coef & expo==> 3
2
Enter coef & expo==> 5
3
The polynomial is==>4x^(1) + 3x^(2) + 5x^(3) Addition of polynomial==>
The polynomial is==>9x^(1) + 5x^(2) + 9x^(3) Enter your choice==>2
Enter no of terms of polynomial==>3
 Enter coef & expo==>3
1
Enter coef & expo==> 4
2
Enter coef & expo==> 5
4
The polynomial is==>3x^(1) + 4x^(2) + 5x^(4)

Enter the value of x=2 Value of polynomial=102 Enter your choice==>3 exit
```

Binary Search Tree (BST)

Exp No 10

AIM:

Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers

a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2

b. Traverse the BST in Inorder, Preorder and Post Order

c. Search the BST for a given element (KEY) and report the appropriate message

d. Delete an element(ELEM) from BST

e. Exit

ALGORITHM:

Preorder Traversal

Step 1: Display root information
Step2: Traverse left sub tree in preorder
Step 3: Traverse right sub tree in preorder

In order Traversal

Step 1: Traverse the left sub tree in order
Step 2: Display root information
Step3: Traverse right sub tree in order

Post order Traversal

Step 1: traverse the left sub tree in post order
Step 2: traverse the right sub tree in post order
Step 3: Display root information

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
struct BST
{
        int data;
        struct BST *left;
        struct BST *right;
};
typedef struct BST NODE;
NODE *node;

NODE* createtree(NODE *node, int data)
{
        if (node == NULL)
        {
                NODE *temp;
                temp= (NODE*)malloc(sizeof(NODE));
                temp->data = data;
                temp->left = temp->right = NULL;
                return temp;
        }
        if (data < (node->data))
        {
                node->left = createtree(node->left, data);
        }
        else if (data > node->data)
        {
                node -> right = createtree(node->right, data);

        }
        return node;
}
NODE* search(NODE *node, int data)
{
        if(node == NULL)
                printf("\nElement not found");
        else if(data < node->data)
        {
                node->left=search(node->left, data);
        }
        else if(data > node->data)
        {
                node->right=search(node->right, data);

        }
        else
        printf("\nElement found is: %d", node->data);
        return node;
}

void inorder(NODE *node)
{
        if(node != NULL)
        {
                inorder(node->left);
                printf("%d\t", node->data);
                inorder(node->right);
        }
}

void preorder(NODE *node)
{
        if(node != NULL)
        {
                printf("%d\t", node->data);
                preorder(node->left);
                preorder(node->right);
        }
}
void postorder(NODE *node)
{
        if(node != NULL)
        {
                postorder(node->left);
                postorder(node->right);
                printf("%d\t", node->data);
        }

}

NODE* findMin(NODE *node)
{
        if(node==NULL)
```

```
    {
        return NULL;
    }
    if(node->left)
        return findMin(node->left);
    else
        return node;
}

NODE* del(NODE *node, int data)
{
    NODE *temp;
    if(node == NULL)
    {
        printf("\nElement not found");
    }
    else if(data < node->data)
    {
        node->left = del(node->left, data);
    }
    else if(data > node->data)
    {
        node->right = del(node->right, data);
    }
    else
    {
        if(node->right && node->left)
        {
            temp = findMin(node->right);
            node -> data = temp->data;
            node -> right = del(node->right,temp->data);

        }
        else
        {
            temp = node;
            if(node->left == NULL)
                node = node->right;
            else if(node->right == NULL)
                node = node->left;
            free(temp);
        }
    }

    return node;
}

void main()
{
    int data, ch, i, n;

    NODE *root=NULL;
    clrscr();
    while (1)
    {
        printf("\n1.Insertion in Binary Search Tree"); printf("\n2.Search Element in Binary Search
Tree");
        printf("\n3.Delete Element in Binary Search Tree");
        printf("\n4.Inorder\n5.Preorder\n6.Postorder\n7.Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:printf("\nEnter N value: " );
                scanf("%d", &n);
                printf("\nEnter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2)\n");
                for(i=0; i<n; i++)
                {
                    scanf("%d", &data);
                    root=createtree(root, data);
                }
            break;

            case 2:printf("\nEnter the element to search: ");
                scanf("%d", &data);
                root=search(root, data);
            break;

            case 3:printf("\nEnter the element to delete: ");
                scanf("%d", &data);
                root=del(root, data);
            break;

            case 4:printf("\nInorder Traversal: \n");
                inorder(root);
            break;

            case 5:printf("\nPreorder Traversal: \n");
                preorder(root);
            break;

            case 6:printf("\nPostorder Traversal: \n");
                postorder(root);
            break;

            case 7:exit(0);

            default : printf("\nWrong option");
            break;

        }
    }
}
```

SAMPLE INPUT AND OUTPUT

```
linux:~/dslab #gedit bst.c linux:~/dslab #gcc bst.c linux:~/dslab # ./a.out

Program For Binary Search Tree 1.Create
2.        Search
3.        Recursive Traversals 4.Exit
Enter your choice :1 Enter The Element 15
Want To enter More Elements?(1/0)1 Enter The Element 25

Want To enter More Elements?(1/0)1 Enter The Element 35

Want To enter More Elements?(1/0)1

Enter The Element 45

Want To enter More Elements?(1/0)1 Enter The Element 5
Want To enter More Elements?(1/0)1 Enter The Element 7
Want To enter More Elements?(1/0)0 Enter your choice :2
Enter Element to be searched :7

The 7 Element is Present Parent of node 7 is 5
1.      Create
2.        Search
3.        Recursive Traversals 4.Exit

Enter your choice :2
Enter Element to be searched :88 The 88 Element is not Present

Enter your choice :3

The Inorder display : 5 7 15 25 35 45
The Preorder display : 15 5 7 25 35 45
The Postorder display : 7 5 45 35 25 15 Enter your choice :4
```

Graph using DFS method

Exp No 11

AIM:

Design, Develop and Implement a Program in C for the following operations on Graph(G) of Cities

a. Create a Graph of N cities using Adjacency Matrix.

b. Print all the nodes reachable from a given starting node in a digraph using BFS method

c. Check whether a given graph is connected or not using DFS method

ALGORITHM

Step 1: Start.
Step 2: Input the value of N nodes of the graph
Step 3: Create a graph of N nodes using adjacency matrix representation.
Step 3: Print the nodes reachable from the starting node using BFS.
Step 4: Check whether graph is connected or not using DFS.
Step 5: Stop.

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
int a[20][20],q[20],visited[20],reach[10],n,i,j,f=0,r=-1,count=0;
void bfs(int v)
{
        for(i=1;i<=n;i++)
                if(a[v][i] && !visited[i])
                        q[++r]=i;
                if(f<=r)
                {
                        visited[q[f]]=1;
                        bfs(q[f++]);
                }
}

void dfs(int v)
{
        int i; reach[v]=1;
        for(i=1;i<=n;i++)
        {
                if(a[v][i] && !reach[i])
                {
                        printf("\n %d->%d",v,i);
                        count++;
                        dfs(i);
                }
        }

}
void main()
{
        int v, choice;
        printf("\n Enter the number of vertices:");
        scanf("%d",&n);

        for(i=1;i<=n;i++)
        {
                q[i]=0;
                visited[i]=0;
        }
        for(i=1;i<=n-1;i++)
                reach[i]=0;

        printf("\n Enter graph data in matrix form:\n");
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        scanf("%d",&a[i][j]);

        printf("1.BFS\n 2.DFS\n 3.Exit\n");
        scanf("%d",&choice);

        switch(choice)
        {
                case 1:
                        printf("\n Enter the starting vertex:");
                        scanf("%d",&v);
                        bfs(v);
                        if((v<1)||(v>n))
                        {
                                printf("\n Bfs is not possible");
                        }
                        else
                        {
                                printf("\n The nodes which are reachable from %d:\n",v);
                                for(i=1;i<=n;i++)
                                        if(visited[i])
                                                printf("%d\t",i);
                        }
                        break;

                case 2:
                        dfs(1);
                        if(count==n-1)
                                printf("\n Graph is connected");
                        else
                                printf("\n Graph is not connected");
                        break;

                case 3:
                        exit(0);
        }
}
```

SAMPLE INPUT AND OUTPUT

```
linux:~/dslab #gedit bfs.c linux:~/dslab #gcc bfs.c linux:~/dslab # ./a.out

Enter the number of vertices:5 Enter graph data in matrix form: 0 1 0 1 0
1 0 1 0 1
0 1 0 1 0
1 0 1 0 0
0 1 0 0 0
1.        BFS
2.        DFS
3.        Exit 2

1->2
2->3
3->4
2->5
Graph is connectedcsdept Enter the number of vertices:5
Enter graph data in matrix form: 0 1 0 1 0 1 0 1 0 0

0 1 0 1 0
1 0 1 0 0
0 0 0 0 0
1.        BFS
2.        DFS
3.        Exit 2

1->2
2->3
3->4
Graph is not connected
Enter the number of vertices:5

Enter graph data in matrix form: 0 1 1 0 0
0 0 0 1 0
0 0 0 0 0
0 0 1 0 0
0 0 1 0 0
1.        BFS
2.        DFS
3.        Exit 1
Enter the starting vertex:1
The nodes which are reachable from 1: 2 3 4

Enter graph data in matrix form: 0 1 1 0 0
0 0 0 1 0
0 0 0 0 0
0 0 1 0 0
0 0 1 0 0
1.        BFS
2.        DFS
3.        Exit 1
Enter the starting vertex:0 BFS is not possible
```

Implementing hashing technique

Exp No 12

AIM

Given a File of N employee records with a set K of Keys(4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table(HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Design and develop a Program in C that uses Hash function H: K → L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing

ALGORITHM

Step 1: Start

Step 2: Initialize all memory locations with some values to identity as space a[i]=-1

Step 3: Read Employee key value .calculate hash key using remainder method hk<-key%100

Step 4: Inserting Employee record using key

```
Inserting hash dull function
If(count=m)
If space is available for that key
If(H[k]==-1)
H[hk] <-key
If collision occurs, it can be solved using linear probing method.
Checking free space from key to end
for(i=hk+1;i<m;i++)
Checking free space from beginning to key value.
for(i=0;i<hk&& flag==0;i++)
```

Step 5: Display all memory location with index and employee key

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
int create(int);
void display (int[]);
void main()
{
        int a[MAX],num,key,i;
        int ans=1;
        printf(" collision handling by linear probing : \n");
        for (i=0;i<MAX;i++)
        {
                a[i] = -1;
        }
        do
        {
                printf("\n Enter the data");
                scanf("%4d", &num);
                key=create(num);
                linear_prob(a,key,num);
                printf("\n Do you wish to continue ? (1/0) ");
        scanf("%d",&ans);
    }while(ans);
    display(a);
}

int create(int num)
{
    int key;
    key=num%100;
    return key;
}

void linear_prob(int a[MAX], int key, int num)
{
    int flag, i, count=0;
    flag=0;
    if(a[key]== -1)
    {
        a[key] = num;
    }
    else
    {
        printf("\nCollision Detected...!!!\n");
        i=0;
        while(i<MAX)
        {
            if (a[i]!=-1)
                count++;
            i++;
        }
        printf("Collision avoided successfully using LINEAR PROBING\n");

        if(count == MAX)
        {
        printf("\n Hash table is full");
        display(a);
        exit(1);

        }
        for(i=key+1; i<MAX; i++)
            if(a[i] == -1)
            {

                a[i] = num;
                flag =1;
                    break;

            }

        i=0;
        while((i<key) && (flag==0))
        {
            if(a[i] == -1)
            {

                a[i] = num;
                flag=1;
                break;
            }
            i++;
        }
    }

}
```

```
void display(int a[MAX])
{
        int i,choice;
        printf("1.Display ALL\n 2.Filtered Display\n");
        scanf("%d",&choice);
        if(choice==1)
        {
                printf("\n the hash table is\n");
                for(i=0; i<MAX; i++)
                        printf("\n %d %d ", i, a[i]);
        }
        else
        {
                printf("\n the hash table is\n");
                for(i=0; i<MAX; i++)
                        if(a[i]!=-1)
                        {
                                printf("\n %d %d ", i, a[i]);
                                continue;
                        }

        }
}
```

SAMPLE INPUT AND OUTPUT

```
linux:~/dslab #gedit hash.c linux:~/dslab #gcc hash.c linux:~/dslab #
./a.out collision handling by linear probing : Enter the data1234

Do you wish to continue ? (1/0) 1 Enter the data2548
Do you wish to continue ? (1/0) 1 Enter the data3256
Do you wish to continue ? (1/0) 1 Enter the data1299
Do you wish to continue ? (1/0) 1 Enter the data1298
Do you wish to continue ? (1/0) 1 Enter the data1398
Collision Detected...!!!
Collision avoided successfully using LINEAR PROBING

Do you wish to continue ? (1/0) 0 1.Display ALL
2.Filtered Display 2
the hash table is 0 1398
34 1234
48 2548
56 3256
98 1298
99 1299
```