VADODARA INSTITUTE OF ENGINEERING

KOTAMBI

Lab Manual

Data Structures (DS)
(Subject Code: 2130702)
(III Semester CE/IT)

Prepared By:
CE/IT Department

Practical List

1	Introduction to pointers. Call by Value and Call by reference.
2	Introduction to Dynamic Memory Allocation. DMA functions malloc(), calloc(), free() etc.
3	Implement a program for stack that performs following operation using array. a) PUSH b) POP
4	Implement a program for stack that performs following operation using array. PEEP b) CHANGE c) DISPLAY
5	Implement a program to convert infix notation to postfix notation using stack.
6	Write a program to implement simple queue using arrays that performs following operations (a) INSERT (b) DELETE (c) DISPLAY
7	Write a program to implement Circular Queue using arrays that performs following Operations. (a) INSERT (b) DELETE (c) DISPLAY
8	Write a menu driven program to implement following operation on the singly linked list. a) Insert a node at the front of the linked list.
9	Write a menu driven program to implement following operation on the singly linked list. a) Insert a node at the end of the linked list.
10	Write a menu driven program to implement following operation on the singly linked list. a) Delete a first node of the linked list.
11	Write a menu driven program to implement following operation on the singly linked list. a) Delete a node before specified position.
12	Write a menu driven program to implement following operation on the singly linked list. a) Delete a node after specified position
13	Write a program to implement stack using linked list.
14	Write a program to implement queue using linked list.
15	Write a program to implement following operations on the doubly linked list. a) Insert a node at the front of the linked list.
16	Write a program to implement following operations on the doubly linked list. a) Insert a node at the end of the linked list.
17	Write a program to implement following operations on the doubly linked list. a) Delete a last node of the linked list.
18	Write a program to implement following operations on the doubly linked list. a) Delete a node before specified position.
19	Write a program to implement following operations on the circular linked list. a) Insert a node at the end of the linked list.
20	Write a program to implement following operations on the circular linked list. a) Insert a node before specified position.
21	Write a program to implement following operations on the circular linked list.

	a) Delete a first node.
22	Write a program to implement following operations on the circular linked list. a) Delete a node after specified position.
23	Write a program which create binary search tree
24	Implement recursive and non-recursive tree traversing method of inorder traversal.
25	Implement recursive and non-recursive tree traversing method of preorder traversal.
26	Write a program to implement Merge Sort.
27	Write a program to implement Bubble Sort.
28	Write a program to implement Binary Search.

List of Equipments and components for A Batch of 20 students (1 per batch)

1. SOFTWARE REQUIRED – **TURBOC version 3**.
2. OPERATING SYSTEM – **WINDOWS 2000 / XP / NT**
3. COMPUTERS SPECIFICATION– **20 Nos**. (Minimum Requirement : Pentium III or Pentium IV with 1GB RAM and 40 GB harddisk)

Practical:1

AIM: Introduction to pointers. Call by Value and Call by reference.

OBJECTIVE:

Pointer:

A **pointer** is a variable whose value is the address of another variable.

i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before you can use it to store any variable address. The general form of a pointer variable declaration is:

type *var-name;

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void swap(int*num1,int*num2)
{
        int temp ;
          temp  =*num1 ;
        *num1 =*num2 ;
        *num2 = temp ;
}
void swapp(int num1,int num2)
{
        int temp;
        temp=num1;
        num1=num2;
        num2=temp;
}

void main()
{
        int num1,num2;
         clrscr();
        printf("\nEnter two numbers no.1 and no.2 : ");
        scanf("%d %d",&num1,&num2);
```

```
        printf("\nbefore swapping");
        printf("\nNo. 1 : %d",num1);
        printf("\nNo. 2 : %d",num2);
        printf(" \nafter swapping, CALL BY VALUE");
        swapp(num1,num2);
        printf("\nafter swapping");
        printf("\nNo. 1 : %d",num1);
        printf("\nNo. 2 : %d",num2);
        printf(" \nafter swapping, CALL BYREFERENCE");
        swap(&num1,&num2);
        printf("\nafter swapping");
        printf("\nNo. 1 : %d",num1);
        printf("\nNo. 2 : %d",num2);

        getch();

}
```

Result:

```
Enter two numbers no.1 and no.2 :  10          20

before swapping
No. 1 : 10
No. 2 : 20
after swapping, CALL BY VALUE
after swapping
No. 1 : 10
No. 2 : 20
after swapping, CALL BYREFERENCE
after swapping
No. 1 : 20
No. 2 : 10
```

Practical:2

AIM: Introduction to Dynamic Memory Allocation. DMA functions malloc(), calloc(), free() etc.

OBJECTIVE:

- **Dynamic memory allocation** is assigning memory locations to variables during execution of the program by explicit request of the programmer. Dynamic allocation is a unique feature to C (amongst high level languages). It enables us to create data types and structures of any size and length to suit our programs need within the program.
- For example, to use arrays, dynamic memory allocation and use, eliminating the need to determine the size of the array at declaration time.
- The following functions are used in c for purpose of memory management.

malloc()	Allocates requested size of bytes and returns a pointer first byte of allocated space
calloc()	Allocates space for an array elements, initializes to zero and then returns a pointer to memory
free()	dellocate the previously allocated space
realloc()	Change the size of previously allocated space

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
void main()
{
 Char *ma;
Clrscr();
//allocating memory space.
ma = malloc(sizeof(char));
ma=”hello”;
strcat(ma,”VIE,”);
printf(“dynamically allocated value: %s\n”ma);
// reallocating memory space.

ma = reallocate(ma,100*sizeof(char));
strcat(ma,“kotambi,vadodara”);
   printf(“\n reallocated value : %s”,ma);
free();
getch();
}
```

Result:

Dynamically allocated value : hello VIE,
Reallocated value: hello VIE,kotambi,vadodara

Practical:3

AIM: Implement a program for stack that performs following operations using array: (a) PUSH (b) POP

PROGRAM:

```
/*PROGRAM TO PERFORM PUSH OPERATION ON STACK USING ARRAY*/
#include<stdio.h>
#include<stdlib.h>

#define MAX 5    //Maximum number of elements that can be stored

int top=-1,stack[MAX];
void push();
void pop();


void main()
{
   int ch;

   while(1)    //infinite loop, will end when choice will be 4
   {
      printf("\n*** Stack Menu ***");
      printf("\n\n1.Push\n2.Pop\n3.Exit");
      printf("\n\nEnter your choice(1-3):");
      scanf("%d",&ch);

      switch(ch)
      {
         case 1: push();
                 break;
         case 2: pop();
                 break;

         case 3: exit(0);

         default: printf("\nWrong Choice!!");
      }
   }
}
```

```
void push()
{
   int val;

   if(top==MAX-1)
   {
      printf("\nStack is full!!");
   }
   else
   {
      printf("\nEnter element to push:");
      scanf("%d",&val);
      top=top+1;
      stack[top]=val;
   }
}

void pop()
{
   if(top==-1)
   {
      printf("\nStack is empty!!");
   }
   else
   {
      printf("\nDeleted element is %d",stack[top]);
      top=top-1;
   }
}
void display()
{
   int i;

   if(top==-1)
   {
      printf("\nStack is empty!!");
   }
   else
   {
      printf("\nStack is...\n");
      for(i=top;i>=0;--i)
         printf("%d\n",stack[i]);
   }
}
```

RESULT:

```
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):1
Enter element to push:3

*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):1
Enter element to push:6

*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):3
Stack is…
6
3
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):2
Deleted element is 6
```

Practical:4

AIM: Implement a program for stack that performs following operations using array: a) PEEP b) CHANGE c) DISPLAY

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#define max 5
void main()
{
int stack[max];
int top=-1;
int choice;
int ele,temp,pos,j,i;
clrscr();
main:
        printf("\nEnter your Choice- 1.push 2.pop 6.Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
            printf("Enter element to be pushed into Stack- \n");
                    scanf("%d",&ele);
                     if(top==max-1)
                        {
                            printf("\nstack overflow \n");
                        }
                      else
                        {
                            top=top++;
                            stack[top]=ele;
                            printf("\nNumber pushed is
%d",ele);
                        }
                    goto main;
                      break;
        case 2:
                    if(top==-1)
                        {
                            printf("\nstack underflow \n");
                        }
```

```
                                        else
                                        {
                                                temp=stack[top];
                                                top--;
                                                printf("\nvalue of deleted element is
%d \n",temp);
                                        }
                                    goto main;
                                      break;


                case 6:
                                      exit();
                default:
                                printf("invalid choice \n");
                                      break;
                }
for(i=top;i>=0;i--)
                                        {
                                                printf("\n %d",stack[i]);
                                        }

getch();
}
```

Result:

```
Enter your choice 1.push  2.pop 6.exit
1
Enter element to be push into stack
1
Number pushed is 1

Enter your choice 1.push  2.pop 6.exit
1
Enter element to be push into stack
2
Number pushed is 2

Enter your choice 1.push  2.pop 6.exit
1
Enter element to be push into stack
3
Number pushed is 3


Enter your choice 1.push  2.pop  6.exit
2
Value of deleted element is 3.

Enter your choice 1.push  2.pop 6.exit
6

Elements in stack are
2
1
```

Practical:5

AIM: Implement a program to convert infix notation to postfix notation using stack.

PROGRAM:

```
#include<conio.h>
#include<stdio.h>
#define max 50
void infixtopostfix();
void push(char);
char pop();
char infix[max],postfix[max],s[max];
int i,j,top=-1;


void main()
{
char c;
clrscr();
l1:
printf("enter string\n");
fflush(stdin);
gets(infix);
infixtopostfix();
printf("string is: ");
puts(postfix);
printf("wnat to cotinue ? y/n:  ");
scanf("%c",&c);
if(c=='y')
{       goto l1;
}
else
{
        exit();
}
getch();
}

void infixtopostfix()
{
```

```
    char temp;
i=0,j=0;
while(infix[i]!='\0')
{
        if(infix[i]=='(')
        {
                push(infix[i]);
                i++;
        }

        else if(infix[i]==')')
        {
                while((top!=-1)&&(s[top]!='('))
                {
                        postfix[j]=pop();
                        j++;
                }
                temp=pop();
                i++;
        }

        else if(isdigit(infix[i])||(isalpha(infix[i])))
        {
                postfix[j]=infix[i];
                i++;
                j++;
        }

        else if(infix[i]=='$'||infix[i]=='^'||infix[i]=='+'||infix[i]=='-
'||infix[i]=='%'||infix[i]=='/'||infix[i]=='*')
        {
                 if(s[top]=='^' && infix[i]=='^')
                 {
                        push(infix[i]);
                        i++;
                 }
                  while(top!=-1 && s[top]!='(' && (priority(s[top])>=priority(infix[i])))
                  {
                        postfix[j]=pop();
                        j++;
                  }
                  push(infix[i]);
                  i++;
        }
        else
        {
```

```
                printf("incorrect");
            exit();
        }
    }
    while(top!=-1 && s[top]!='(')
    {
        postfix[j]=pop();
        j++;
    }
}
void push(char a)
{
    if(top==max-1)
    {
        printf("stack overflow \n");
    }
    else
    {
        top++;
        s[top]=a;
    }
}
char pop()
{
    char val=' ';
    if(top==-1)
    {
        printf("stack empty \n");
    }
    else
    {
        val=s[top];
        top--;
    }
    return val;
}
int priority(char p)
{
    if(p=='$'||p=='^')
    {
        return 2;
    }
    else if(p=='/'||p=='%'||p=='*')
    {
        return 1;
    }
```

```
        else if(p=='+'||p=='-')
    {
            return 0;
    }   }
```

RESULT:

Enter your Choice - 1

Enter element to be pushed into Stack- 3

Number pushed is 3

Enter your Choice- 1

Enter element to be pushed into Stack- 5

Number pushed is 5

Enter your Choice- 1

Enter element to be pushed into Stack- 4

Number pushed is 4

Enter your Choice – 2

enter position-1

element is- 5

Enter you choice
6
Elements in stack are:-
3
2
1

Practical:6

AIM Write a program to implement simple queue using arrays that performs following operations (a) INSERT (b) DELETE (c) DISPLAY

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#define max 5
int q[10],front=0,rear=-1;
void main()
{
   int ch;
   void insert();
   void delet();
   void display();
   clrscr();
   printf("\nQueue operations\n");
   printf("1.insert\n2.delete\n3.display\n4.exit\n");
   while(1)
   {
      printf("Enter your choice:");
      scanf("%d",&ch);
      switch(ch)
      {
      case 1: insert();
         break;
      case 2: delet();
         break;
      case 3:display();
         break;
      case 4:exit();
      default:printf("Invalid option\n");
      }
   }
}

void insert()
{
   int x;
   if(rear==max-1)
      printf("Queue is overflow\n");
   else
   {
      printf("Enter element to be insert:");
      scanf("%d",&x);
```

```
        q[++rear]=x;
    }
}
void  delet()
{
    int a;
    if((front==0)&&(rear==-1))
    {
        printf("Queue is underflow\n");
        getch();
        exit();
    }
    a=q[front++];
    printf("Deleted element is:%d\n",a);
    if(front>rear)
    {
        front=0;
        rear=-1;
    }
}

void display()
{
    int i;
    if(front==0&&rear==-1)
    {
        printf("Queue is underflow\n");
        getch();
        exit();
    }
    for(i=front;i<=rear;i++)
        printf("\t%d",q[i]);
    printf("\n");
}
getch();
```

Result :-

```
Queue operations
1.insert
2.delete
3.display
4.exit
Enter your choice:1
Enter element to be insert:56
```

```
Enter your choice:1
Enter element to be insert:25

Enter your choice:1
Enter element to be insert:75

Enter your choice:2
Deleted element is: 56

Enter your choice:3
25
75
```

Practical:7

AIM: Write a program to implement Circular Queue using arrays that performs following Operations. (a) INSERT (b) DELETE (c) DISPLAY

PROGRAM:

```
#include<stdio.h>
#define max 3
int q[max],front=0,rear=-1;
void main()
{
   int ch;
   void insert();
   void delet();
   void display();
   clrscr();
   printf("\nCircular Queue operations\n");
   printf("1.insert\n2.delete\n3.display\n4.exit\n");
   while(1)
   {
      printf("Enter your choice:");
      scanf("%d",&ch);
      switch(ch)
      {
      case 1: insert();
         break;
      case 2: delet();
         break;
      case 3:display();
         break;
      case 4:exit();
      default:printf("Invalid option\n");
      }
   }
}

void insert()
{
   int x;
   if((front==0&&rear==max-1)||(front>0&&rear==front-1))
      printf("Queue is overflow\n");
   else
   {
      printf("Enter element to be insert:");
      scanf("%d",&x);
      if(rear==max-1&&front>0)
```

```
            {
            rear=0;
            q[rear]=x;
        }
        else
        {
            if((front==0&&rear==-1)||(rear!=front-1))
                q[++rear]=x;
        }
    }
}
void  delet()
{
    int a;
    if((front==0)&&(rear==-1))
    {
        printf("Queue is underflow\n");
        getch();
        exit();
    }
    if(front==rear)
    {
        a=q[front];
        rear=-1;
        front=0;
    }
    else
        if(front==max-1)
        {
            a=q[front];
            front=0;
        }
        else a=q[front++];
        printf("Deleted element is:%d\n",a);
}

void display()
{
    int i,j;
    if(front==0&&rear==-1)
    {
        printf("Queue is underflow\n");
        getch();
        exit();
    }
    if(front>rear)
```

```
        {
      for(i=0;i<=rear;i++)
         printf("\t%d",q[i]);
      for(j=front;j<=max-1;j++)
         printf("\t%d",q[j]);
      printf("\nrear is at %d\n",q[rear]);
      printf("\nfront is at %d\n",q[front]);
   }
   else
   {
      for(i=front;i<=rear;i++)
      {
         printf("\t%d",q[i]);
      }
      printf("\nrear is at %d\n",q[rear]);
      printf("\nfront is at %d\n",q[front]);
   }
   printf("\n");
}
getch();
```

Output:

```
Circular Queue operations
1.insert
2.delete
3.display
4.exit

Enter your choice:1
Enter element to be insert:45

Enter your choice:1
Enter element to be insert:50

Enter your choice:1
Enter element to be insert:10

Enter your choice:2
Deleted element is: 45

Enter your choice:3
50
10
```

Practical:8

AIM: Write a menu driven program to implement following operation on the singly linked list. a) Insert a node at the front of the linked list.

PROGRAM:-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node * insert_beg();
void display();
struct node
{
        int data;
        struct node *next;

};
struct node *start=NULL;

void main()
{
        int ch;
        clrscr();
        while(1)
        {
                printf("\n ***LINKLIST MENU***");
                printf("\n\n1.insert_beg\n2.display\n3.exit");
                printf("\n\n enter your choice (1 2 or 3)");
                scanf("%d",&ch);

                switch(ch)
                {
                        case 1:start=insert_beg();
                             break;
                        case 2:display();break;
                        case 3:exit(0);


                        default:printf("\nwrong coice!");
                                break;   }
        }
```

```
        getch();
}
struct node * insert_beg()
{
      struct node *new_node;
      int val;
      new_node=(struct node*)(malloc(sizeof(struct node)));
      printf("Enter an element:");
      scanf("%d",&val);

      new_node->data=val;
      new_node->next=start;
      start=new_node;

      return start;}

void display()
{      struct node *ptr;
         ptr=start;
         while(ptr!=NULL)
         {
                  printf("\nelement is %d",ptr->data);
                  ptr=ptr->next;
         }
}
```

Output:

```
***LINKLIST MENU***
1.insert_beg
2.display
3.exit
enter your choice (1 2 or 3): 1
enter an element:12

enter your choice (1 2 or 3): 1
enter an element:24

enter your choice (1 2 or 3): 1
enter an element:48

enter your choice (1 2 or 3): 2
element is 48
element is 24
element is 12
```

Practical:9

AIM: I. **Write a menu driven program to implement following operation on the singly linked list. a) Insert a node at the end of the linked list.**

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node * insert_beg();
struct node * insert_end();
void display();
struct node
{int data;
struct node *next;
};
struct node *start=NULL;

void main()
{
        int ch;
        clrscr();
        while(1)
        {
                printf("\n ***LINKLIST MENU***");
                printf("\n\n1.insert_beg\n2. insert_end \n 3. display\n4.exit");
                printf("\n\n enter your choice (1 2 3 or 4)");
                scanf("%d",&ch);

                switch(ch)
                {
                        case 1:start=insert_beg();
                            break;
                        case 2:start=insert_end();
                            break;
                        case 3:display();
                            break;
                        case 4:exit(0);
                                break;
                        default:printf("\nwrong coice!");
                                break;
                }
        }
```

```
                getch();}
struct node * insert_beg()
{
      struct node *new_node;

      int val;
      new_node=(struct node*)(malloc(sizeof(struct node)));
      printf("Enter an element:");
      scanf("%d",&val);

      new_node->data=val;
      new_node->next=start;
      start=new_node;

      return start;}


struct node * insert_end()
{
          struct node *new_node,*ptr;

      int val,i=1;
      new_node=(struct node*)(malloc(sizeof(struct node)));
      printf("Enter an element:");
      scanf("%d",&val);

      new_node->data=val;
      new_node->next=NULL;


      ptr=start;
      if(start==NULL)       //if link list is empty
      {             start=new_node;}
      else
      {
                    while(ptr->next!=NULL)
                    {                   ptr=ptr->next;    }

                    ptr->next=new_node;
      }
          return start;   }


void display()
{       struct node *ptr;
          ptr=start;
```

```
        while(ptr!=NULL)
    {
            printf("\nelement is %d",ptr->data);
            ptr=ptr->next;
    }  }
```

Output:

```
***LINKLIST MENU***
1.insert_beg
2. insert_end
3.display
3.exit

enter your choice (1 2  3 or 4): 1
enter an element:12

enter your choice (1 2 3 or 4): 2
enter an element:24

enter your choice (1 2 3 or 4): 2
enter an element:48

enter your choice (1 2 3 or 4): 3
element is 12
element is 24
element is 48
```

Practical:10

AIM: Write a menu driven program to implement following operation on the singly linked list. a) Delete a first node of the linked list.

OBJECTIVE:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node * insert_beg();
struct node * insert_end();
struct node * delete_front();
void display();
struct node
{     int data;
      struct node *next;
};
struct node *start=NULL;
void main()
{     int ch;
      clrscr();
      while(1)
      {     printf("\n ***LINKLIST MENU***");
            printf("\n\n1.insert_beg\n2. insert_end \n 3.delete_front\n 4. display\n5.exit");
            printf("\n\n enter your choice ");
            scanf("%d",&ch);
      switch(ch)
            {
                  case 1:start=insert_beg();
                       break;
                  case 2:start=insert_end();
```

```
                    break;
                case 3:start=delete_front();
                    break;
                case 4 :    display();
                    break;
                case 5:exit(0);
                    break;
                default:printf("\nwrong coice!");
                    break;          }
        }   }
struct node * insert_beg()
{
    struct node *new_node;
    int val;
    new_node=(struct node*)(malloc(sizeof(struct node)));
    printf("\t\t\t\tEnter an element:");
    scanf("%d",&val);
    new_node->data=val;
    new_node->next=start;
    start=new_node;
    return start;}
struct node * insert_end()
{
     struct node *new_node,*ptr;
    int val;
    new_node=(struct node*)(malloc(sizeof(struct node)));
    printf("Enter an element:");
    scanf("%d",&val);
    new_node->data=val;
    new_node->next=NULL;
    ptr=start;
```

```
        if(start==NULL)      //if link list is empty
    {           start=new_node;      }
    else
    {           while(ptr->next!=NULL)
            {           ptr=ptr->next;    }
            ptr->next=new_node;}
     return start;}
struct node * delete_front()
{
     //If the list is already empty
            struct node *new_node,*ptr;
     if(start== NULL)
     {
            printf("\nEmpty Linked List. Deletion not possible.\n");
     }
     else
     {
            ptr = start;
            start ->next= ptr->next;
            free(ptr);
            printf("\nNode deleted from the front.\n");
     }
     return start;
}

void display()
{     struct node *ptr;
     ptr=start;
     while(ptr!=NULL)
     {
            printf("\nelement is %d",ptr->data);
```

```
ptr=ptr->next;}}
```

Output:

```
***LINKLIST MENU***
1.insert_beg
2.insert_end
3.delete_front
4.display
5.exit

enter your choice : 1
enter an element:12

enter your choice : 2
enter an element:24

enter your choice : 2
enter an element:55

enter your choice : 3
Node deleted from the front.

enter your choice 4
element is 24
element is 55
```

Practical:11

AIM: Write a menu driven program to implement following operation on the singly linked list. a) Delete a node at specified position.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
 struct Node
{
   int data;
   struct Node *next;
};

void push(struct Node** head_ref, int new_data)
{
   struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
   new_node->data  = new_data;
   new_node->next = (*head_ref);
   (*head_ref)    = new_node;
}

void deleteNode(struct Node **head_ref, int position)
{
  // If linked list is empty
  if (*head_ref == NULL)
     return;

  // Store head node
  struct Node* temp = *head_ref;

   // If head needs to be removed
   if (position == 0)
   {
      *head_ref = temp->next;   // Change head
      free(temp);               // free old head
      return;
   }

   // Find previous node of the node to be deleted
   for (int i=0; temp!=NULL && i<position-1; i++)
       temp = temp->next;

   // If position is more than number of ndoes
   if (temp == NULL || temp->next == NULL)
```

```
        return;

    // Node temp->next is the node to be deleted
    // Store pointer to the next of node to be deleted
    struct Node *next = temp->next->next;

    // Unlink the node from linked list
    free(temp->next);  // Free memory

    temp->next = next;  // Unlink the deleted node from list
}

// This function prints contents of linked list starting from
// the given node
void printList(struct Node *node)
{
    while (node != NULL)
    {
        printf(" %d ", node->data);
        node = node->next;
    }
}

/* Drier program to test above functions*/
int main()
{
    /* Start with the empty list */
    struct Node* head = NULL;

    push(&head, 7);
    push(&head, 1);
    push(&head, 3);
    push(&head, 2);
    push(&head, 8);

    puts("Created Linked List: ");
    printList(head);
    deleteNode(&head, 4);
    puts("\nLinked List after Deletion at position 4: ");
    printList(head);
    return 0;
}
```

Output

```
Created Linked List:
 8  2  3  1  7
Linked List after Deletion at position 4:
 8  2  3  1
```

Practical:12

AIM: Write a menu driven program to implement following operation on the singly linked list. a) Delete a node after specified position.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
struct node
{
        int data;
        struct node *link;
};

struct node *header, *ptr, *ptr1, *temp;
void insert_end();
void delete_front();
void delete_end();
void delete_any();
void display();

void main()
{
        int choice;
        int cont = 1;
        header = (struct node *) malloc(sizeof(struct node));
       clrscr();

        //Set the content of header node
        header->data = NULL;
        header->link = NULL;

        while(cont == 1)
        {
                printf("\n1. Insert at end\n");
                printf("\n2. Delete from front\n");
                printf("\n3. Delete from end\n");
                printf("\n4. Delete from anywhere\n");
                printf("\n5. Display linked list\n");
                printf("\nEnter your choice: ");
                scanf("%d", &choice);
```

```
                switch(choice)
        {
                case 1:
                        insert_end();
                        break;
                case 2:
                        delete_front();
                        break;
                case 3:
                        delete_end();
                        break;
                case 4:
                        delete_any();
                        break;
                case 5:
                        display();
                        break;
        }

        printf("\n\nDo you want to continue? (1 / 0): ");
        scanf("%d", &cont);    }
getch();
}
void insert_end()
{
        int data_value;

        printf("\nEnter data of the node: ");
        scanf("%d", &data_value);

        temp = (struct node *) malloc(sizeof(struct node));

        //Traverse to the end of the linked list.
        ptr = header;
        while(ptr->link != NULL)
        {
                ptr = ptr->link;
        }

        temp->data = data_value;
        temp->link = ptr->link;
        ptr->link = temp;
}
```

```
//Function to delete a node from the front of a linked list.
void delete_front()
{
        //If the list is already empty
        if(header->link == NULL)
        {
                printf("\nEmpty Linked List. Deletion not possible.\n");
        }
        else
        {
                ptr = header->link;
                header->link= ptr->link;
                free(ptr);
                printf("\nNode deleted from the front.\n");
        }
}

//Function to delete a node from the end of a linked list.
void delete_end()
{
        if(header->link == NULL)
        {
                printf("\nEmpty Linked List. Deletion not possible.\n");
        }
        else
        {
                //Traverse to the end of the list.
                ptr = header;
                while(ptr->link != NULL)
                {
                        ptr1 = ptr;
                        ptr = ptr->link;
                }
                ptr1->link = ptr->link;
                free(ptr);
                printf("\nNode deleted from the end.\n");
        }
}

//Function to delete any node from linked list.
void delete_any()
{
        int key;
```

```
        if(header->link == NULL)
    {
        printf("\nEmpty Linked List. Deletion not possible.\n");
    }
    else
    {
        printf("\nEnter the data of the node to be deleted: ");
        scanf("%d", &key);

        ptr = header;
        while((ptr->link != NULL) && (ptr->data != key))
        {
            ptr1 = ptr;
            ptr = ptr->link;
        }
        if(ptr->data == key)
        {
            ptr1->link = ptr->link;
            free(ptr);
            printf("\nNode with data %d deleted.\n", key);
        }
        else
        {
            printf("\nValue %d not found. Deletion not possible.\n", key);
        }
    }
}

//Function to display the contents of the linked list.
void display()
{
    printf("\nContents of the linked list are: \n");
    //Print the contents of the linked list starting from header
    ptr = header;
    while(ptr->link != NULL)
    {
        ptr = ptr->link;
        printf("%d ", ptr->data);
    }
}
```

Output:

1. Insert at end

2. Delete from front
3. Delete from end
4. Delete from anywhere
5. Display linked list\n

Enter your choice:1
Enter data of the node: 25

Enter your choice:1
Enter data of the node: 55

Enter your choice:1
Enter data of the node: 78

Enter your choice:1
Enter data of the node: 56

Enter your choice:4
nEnter the data of the node to be deleted: 55
Node with data 78 deleted

Enter your choice:5
25
55
56

Practical:13

AIM: Write a program to implement stack using linked list.

PROGRAM:

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

struct node
{
   int info;
   struct node *ptr;
}*top,*top1,*temp;

int topelement();
void push(int data);
void pop();
void display();
void destroy();
void stack_count();
void create();

int count = 0;

void main()
{
   int no, ch, e;

   printf("\n 1 - Push");
   printf("\n 2 - Pop");
   printf("\n 3 - Top");
   printf("\n 4 - Stack Count");
   printf("\n 5 - Dipslay");
   printf("\n 6 - Exit");
   create();

   while (1)
   {
      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:
            if (top == NULL)
                printf("No elements in stack");
            else
            {
                e = topelement();
                printf("\n Top element : %d", e);
            }
            break;
        case 4:
            stack_count();
            break;
         case 5:
            display();
            break;
         case 6:
            exit(0);
        default :
            printf(" Wrong choice, Please enter correct choice  ");
            break;
        }
    }
}

/* Create empty stack */
void create()
{
    top = NULL;
}

/* Count stack elements */
void stack_count()
{
    printf("\n No. of elements in stack : %d", count);
}

/* 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;
    }
    count++;
}

/* Display stack elements */
void display()
{
    top1 = top;

    if (top1 == NULL)
    {
        printf("Stack is empty");
        return;
    }

    while (top1 != NULL)
    {
        printf("%d ", top1->info);
        top1 = top1->ptr;
    }
 }

/* Pop Operation on stack */
void pop()
{
    top1 = top;

    if (top1 == NULL)
    {
        printf("\n Error : Trying to pop from empty stack");
        return;
    }
    else
        top1 = top1->ptr;
    printf("\n Popped value : %d", top->info);
```

```
        free(top);
    top = top1;
    count--;
}

/* Return top element */
int topelement()
{
    return(top->info);
}

/* Check if stack is empty or not */
void empty()
{
    if (top == NULL)
        printf("\n Stack is empty");
    else
        printf("\n Stack is not empty with %d elements", count);
}
```

Result:

```
 1 - Push
 2 - Pop
 3 - Top
 4 - Stack Count
 5 - Dipslay
 6 - Exit
 Enter choice : 1
Enter data : 10

 Enter choice : 1
Enter data : 20

 Enter choice : 1
Enter data : 30

 Enter choice : 5
30 20 10
 Enter choice : 2

 Popped value : 30
 Enter choice : 3
```

```
Top element : 20
Enter choice : 4
```

Practical:14

AIM: Write a program to implement queue using linked list.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
        struct node
        {
                int data;
                struct node *next;
        }       *f=NULL,*r=NULL,*ptr,*newnode;
int ele,a;
void insert();
void delete();
void display();
void main()
{
int x;
clrscr();
hello:
printf("-------QUEUE Menu--------");
printf("\n1.insert \n2.delete \n3.display \n4.exit");
printf("\nenter your choice");
scanf("%d",&x);

switch(x)
{
        case 1: insert();
                goto hello;
        case 2: delete();
                goto hello;
        case 3: display();
                goto hello;
        case 4: exit();

}
getch();
}
void insert()
{
        printf("enter the element");
        scanf("%d",&ele);
```

```
        newnode=(struct node*)malloc(sizeof (struct node));
    newnode->data=ele;
    newnode->next=NULL;
    if(r==NULL)
    {
        r=newnode;
        f=r;
    }
    else
    {
        r->next=newnode;
        r=newnode;
    }
}
void display()
{
    if(f==NULL)
    {
        printf("link list is empty");
    }
    else
    {
        ptr=f;
        while(ptr->next!=NULL)
        {
            printf("%d->",ptr->data);
            ptr=ptr->next;
        }
        printf("%d",ptr->data);
    }
}
void delete()
{
    if(f==NULL)
    {
        printf("linklist is overflow");
    }
    else
    {
        ptr=f;
        f=f->next;
        printf("deleted element is %d",ptr->data);
        free (ptr);
    }

}
```

Output

```
-------QUEUE Menu--------
1.insert
2.delete
3.display
4.exit
enter your choice
1
enter the element11
-------QUEUE Menu--------
1.insert
2.delete
3.display
4.exit
enter your choice1
enter the element22
-------QUEUE Menu--------
1.insert
2.delete
3.display
4.exit
enter your choice1
enter the element33
-------QUEUE Menu--------
1.insert
2.delete
```

Practical:15

AIM: Write a program to implement following operations on the doubly linked list.
a) Insert a node at the front of the linked list.

```
#include<conio.h>
#include<stdio.h>
#include<malloc.h>
#include<process.h>
struct node
   {
int num;
struct node *next;
struct node *prev;
};
struct node *head=NULL,*temp, *first, *last;
int info;
void display();
void insert_at_begin();

void main()
{
int i;
clrscr();
printf("\nprogram for insertion in a doubly linked list :\n");
do
{
printf("\nEnter your choice :\n");
printf("\n1.Insert element at the end of the linkedlist :");
printf("\n2.display");
printf("\n4.Exit\n");
fflush(stdin);
scanf("%d",&i);
switch(i)
{
case 1:
insert_at_begin();
display();
break;

case 2:
//insert_at_specifiedpos();
display();
```

```
 break;
case 4:
exit(0);
}
}
while(i<=4);
getch();
}
void display()
{

struct node *ptr;
ptr=head;
printf("\nStatus of the doubly linked list is as follows :\n");
while(ptr!=NULL)                /* traversing the linked list */
{
printf("\n%d",ptr->num);
ptr=ptr->next;
}
}
void insert_at_begin()
{
printf("\nEnter the value which do you want to insert at begining\n");
scanf("%d",&info);
temp=(struct node *)malloc(sizeof(struct node));//(struct node)malloc(sizeof(NODE));
temp->num=info;
temp->next=NULL;
temp->prev=NULL;
if(head==NULL)
{
head=temp;
last=temp;
}
else
{
temp->next=head;
head->prev=temp;
temp->prev=NULL;
head=temp;
}
}
```

Output

```
Enter your choice:-
1.insert element at front
2.display
```

3.exit

1
Entr the element which u want to insert at beginning
10

Enter your choice:-
1.insert element at front
2.display
3.exit

1

Entr the element which u want to insert at beginning
20

Enter your choice:-
1.insert element at front
2.display
3.exit

1
Entr the element which u want to insert at beginning
30

Enter your choice:-
1.insert element at front
2.display
3.exit
2
Status of dubly linked list is:-
30
20
10

Practical:16

AIM: Write a program to implement following operations on the doubly linked list.
a) Insert a node at the end of the linked list.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
#include<process.h>
struct node
   {
int num;
struct node *next;
struct node *prev;
};
struct node *head=NULL,*temp, *first, *last;/* declaring a global node of type struct */
//typedef struct node NODE;   /* providing a type definition to the above created
structure */
//NODE *head=NULL; /* declaring some of the global variable that would be used
throughout the program */
//NODE *temp, *first, last;
int info;
void display();
//void insert_at_end();
void insert_at_end();
//void insert_at_specifiedpos();
void main()          /* starting the main method() */
{
int i;
clrscr();
printf("\nprogram for insertion in a doubly linked list :\n");
do
{
printf("\nEnter your choice :\n");
printf("\n1.Insert element at the end of the linkedlist :");
printf("\n2.display");
printf("\n4.Exit\n");
fflush(stdin);
scanf("%d",&i);
switch(i)
{
case 1:
```

```
insert_at_end();
display();
break;

case 2:
//insert_at_specifiedpos();
display();
break;
case 4:
exit(0);
}
}
while(i<=4);
getch();
}
void display()
{

struct node *ptr;
ptr=head;
printf("\nStatus of the doubly linked list is as follows :\n");
while(ptr!=NULL)                /* traversing the linked list */
{
printf("\n%d",ptr->num);
ptr=ptr->next;
}
}
void insert_at_end()
{
struct node *ptr;
printf("\nEnter your element in the linked list :");
scanf("%d",&info);
temp=(struct node *)malloc(sizeof(struct node));   /* allocating memory for the node to
be inserted */
temp->num=info;
temp->next=NULL;
temp->prev=NULL;
if(head==NULL)
{
head=temp;
last=temp;
}
ptr=head;
while(ptr->next!=NULL)
{
ptr=ptr->next;
```

```
    }
ptr->next=temp;
temp->prev=ptr;
temp->next=NULL;

}  }  }
```

Output

```
Enter your choice:-
1.insert element at end
2.display
3.exit

Enter element in linked list
10

Enter your choice:-
1.insert element at end
2.display
3.exit


Enter element in linked list
20
Enter your choice:-
1.insert element at end
2.display
3.exit


Enter element in linked list
30
Enter your choice:-
1.insert element at end
2.display
3.exit


Status of doubly linked list
10
20
30
```

Practical:17

AIM: Write a program to implement following operations on the doubly linked list.
Delete a last node of the linked list.

PROGRAM:

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

/* a node of the doubly linked list */
struct Node
{
int data;
struct Node *next;
struct Node *prev;
};

/* Function to delete a node in a Doubly Linked List.
head_ref --> pointer to head node pointer.
del --> pointer to node to be deleted. */
void deleteNode(struct Node **head_ref, struct Node *del)
{
/* base case */
if(*head_ref == NULL || del == NULL)
        return;

/* If node to be deleted is head node */
if(*head_ref == del)
        *head_ref = del->next;

/* Change next only if node to be deleted is NOT the last node */
if(del->next != NULL)
        del->next->prev = del->prev;

/* Change prev only if node to be deleted is NOT the first node */
if(del->prev != NULL)
        del->prev->next = del->next;

/* Finally, free the memory occupied by del*/
free(del);
return;
}

/* UTILITY FUNCTIONS */
```

```
/* Function to insert a node at the beginning of the Doubly Linked List */
void push(struct Node** head_ref, int new_data)
{
/* allocate node */
struct Node* new_node =
        (struct Node*) malloc(sizeof(struct Node));

/* put in the data */
new_node->data = new_data;

/* since we are adding at the begining,
        prev is always NULL */
new_node->prev = NULL;

/* link the old list off the new node */
new_node->next = (*head_ref);

/* change prev of head node to new node */
if((*head_ref) != NULL)
(*head_ref)->prev = new_node ;

/* move the head to point to the new node */
(*head_ref) = new_node;
}

/* Function to print nodes in a given doubly linked list
This function is same as printList() of singly linked lsit */
void printList(struct Node *node)
{
while(node!=NULL)
{
printf("%d ", node->data);
node = node->next;
}
}

/* Drier program to test above functions*/
int main()
{
/* Start with the empty list */
struct Node* head = NULL;

/* Let us create the doubly linked list 10<->8<->4<->2 */
push(&head, 2);
push(&head, 4);
push(&head, 8);
```

```
push(&head, 10);

printf("\n Original Linked list ");
printList(head);

/* delete nodes from the doubly linked list */
deleteNode(&head, head); /*delete first node*/
deleteNode(&head, head->next); /*delete middle node*/
deleteNode(&head, head->next); /*delete last node*/

/* Modified linked list will be NULL<-8->NULL */
printf("\n Modified Linked list ");
printList(head);

getchar();
}
```

Output

Original Linked list 10 8 4 2
Modified Linked list 8

Practical:18

AIM: Write a program to implement following operations on the doubly linked list. Delete a node before specified position.

PROGRAM:

```
#include < stdio.h>
#include < conio.h>
#include < malloc.h>
#include < process.h>
#include < ctype.h>

struct doubly_list
{
   int info;
   struct doubly_list *prev;
   struct doubly_list *next;
}*first,*last,*newnode,*ptr;

void main()
{
   int item,i,loc,loc1;
   char ch;
   clrscr();
   newnode=(struct doubly_list*)malloc(sizeof(struct doubly_list));
   first=newnode;
   last=newnode;
   newnode->prev=NULL;
   do
   {
      printf("\nEnter data: ");
      scanf("%d",&item);
      newnode->info=item;
      printf("\nDo you want to create another node:(y/n)");
      fflush(stdin);
      scanf("%c",&ch);
      if(tolower(ch)=='y')
      {
         newnode->next=(struct doubly_list*)malloc(sizeof(struct doubly_list));
         newnode->next->prev=newnode;
         newnode=newnode->next;
         last=newnode;
      }
      else
      {
         newnode->next=NULL;
      }
   }while(tolower(ch)!='n');
   printf("\nDoubly Linked List is:");
   ptr=first;
   i=1;
   while(ptr!=NULL)
   {
```

```
            printf("\nNode %d : %d",i,ptr->info);
        ptr=ptr->next;
        i++;
    }

    printf("\nEnter the location: ");
    scanf("%d",&loc);
    loc1=1;
    ptr=first;
    while(loc1<loc)
    {
        loc1=loc1+1;
        ptr=ptr->next;
    }
    ptr->prev->next=ptr->next;
    ptr->next->prev=ptr->prev;

printf("\nAfter deletion Doubly Linked List is:");
    ptr=first;
    i=1;
    while(ptr!=NULL)
    {
        printf("\nNode %d : %d",i,ptr->info);
        ptr=ptr->next;
        i++;
    }
    getch();
}
```

Practical:19

AIM: Write a program to implement following operations on the circular linked list. Insert a node at the end of the linked list.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node * insert_beg();
struct node * insert_end();
void display();
struct node
{
        int data;
        struct node *next;

};
struct node *start=NULL;

void main()
{

        int ch;
        clrscr();


        while(1)
        {
                printf("\n ***CIRCULAR LINKLIST MENU***");
                printf("\n\n1.insert_beg\n2. insert_end \n 3.Display\n 4.exit");
                printf("\n\n enter your choice ");
                scanf("%d",&ch);

                switch(ch)
                {
                        case 1:start=insert_beg();
                             break;
                        case 2:start=insert_end();
                             break;

                        case 3:display();
                      break;
```

```
                case 4: exit(0);
                        break;
                default:printf("\nwrong coice!");
                        break;
                }
        }
getch();}


struct node * insert_beg()
{
    struct node *new_node,*ptr;

    int val;
    new_node=(struct node*)(malloc(sizeof(struct node)));
    printf("Enter an element:");
    scanf("%d",&val);

    new_node->data=val;
    ptr=start;

     while(ptr->next!=start)
                {
                ptr=ptr->next;
                }
    new_node->next=start;
    ptr->next=new_node;
    start=new_node;
    return start;  }


struct node * insert_end()
{
        struct node *new_node,*ptr;

    int val;
    new_node=(struct node*)(malloc(sizeof(struct node)));
    printf("Enter an element:");
    scanf("%d",&val);

    new_node->data=val;
    new_node->next=start;


    ptr=start;
```

```
        if(start==NULL)       //if link list is empty
    {
          start=new_node;          }
    else
    {
                while(ptr->next!=start)
                {
                        ptr=ptr->next;
                }
ptr->next=new_node;
    }
       return start;       }
void display()
{      struct node *ptr;
       ptr=start;
       while(ptr!=NULL)
       {   printf("\nelement is %d",ptr->data);
         ptr=ptr->next;
       }   }
```

Output

```
***CIRCULAR LINKLIST MENU***
1..insert_beg
2. insert_end
3.Display
4.exit

enter your choice : 2
Enter an element: 45

enter your choice: 2
Enter an element:23

enter your choice:2
Enter an element:78

enter your choice:3
45
23
78
```

Practical:20

AIM: Write a program to implement following operations on the circular linked list. Insert a node before specified position.

PROGRAM:

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

//Create a basic structure for NODE from which new nodes can be created.
struct node
{
        int data;
        struct node *link;
};

//Initialize 3 pointers as globals so that they do not need to be passed in functions.
struct node *header, *ptr, *temp;

//Prototypes for various user defined functions.
void insert_front();
void insert_end();
void insert_any();
void display();

void main()
{
        int choice;
        int cont = 1;

        //Allocate memory for header node.
        header = (struct node *) malloc(sizeof(struct node));

        clrscr();

        //Set the content of header node
        header->data = NULL;
        header->link = header;

        while(cont == 1)
        {
                //Display menu to the user
                printf("\n1. Insert at front\n");
```

```
                printf("\n2. Insert at end\n");
            printf("\n3. Insert at any position\n");
            printf("\n4. Display linked list\n");
            printf("\nEnter your choice: ");
            scanf("%d", &choice);

            switch(choice)
            {
                case 1:
                    insert_front();
                    break;
                case 2:
                    insert_end();
                    break;
                case 3:
                    insert_any();
                    break;
                case 4:
                    display();
                    break;
            }

            printf("\n\nDo you want to continue? (1 / 0): ");
            scanf("%d", &cont);
        }

        getch();
}

//Function to insert a node at the front of a circular linked list.
void insert_front()
{
        int data_value;

        printf("\nEnter data of the node: ");
        scanf("%d", &data_value);

        temp = (struct node *) malloc(sizeof(struct node));

        temp->data = data_value;
        temp->link = header->link;
        header->link = temp;
}

//Function to insert a node at the end of a circular linked list.
void insert_end()
```

```
    {
        int data_value;

        printf("\nEnter data of the node: ");
        scanf("%d", &data_value);

        temp = (struct node *) malloc(sizeof(struct node));

        //Traverse to the end of the linked list.
        ptr = header;
        while(ptr->link != header)
        {
                ptr = ptr->link;
        }

        temp->data = data_value;
        temp->link = ptr->link;
        ptr->link = temp;
}

//Function to insert a node at any position after a particular node.
void insert_any()
{
        int data_value, key;

        printf("\nEnter data of the node: ");
        scanf("%d", &data_value);
        printf("\nEnter data of the node after which new node is to be inserted: ");
        scanf("%d", &key);

        temp = (struct node *) malloc(sizeof(struct node));

        //Traverse till key is found or end of the linked list is reached.
        ptr = header;
        while(ptr->link != header && ptr->data != key)
        {
                ptr = ptr->link;
        }
        if(ptr->data == key)
        {
                temp->data = data_value;
                temp->link = ptr->link;
                ptr->link = temp;
        }
        else
        {
```

```
                printf("\nValue %d not found\n",key);
        }
}

//Function to display the contents of the linked list.
void display()
{
        printf("\nContents of the linked list are: \n");
        //Print the contents of the linked list starting from header
        ptr = header;
        while(ptr->link != header)
        {
                ptr = ptr->link;
                printf("%d ", ptr->data);
        }
}
```

Output

```
1. Insert at front
2. Insert at end
3. Insert at any position
4. Display linked list

Enter your choice:1
Enter data of the node:45

Enter your choice:1
Enter data of the node:87

Enter your choice:3
Enter data of the node after which new node is to be inserted: 45
Enter data of the node: 12

Enter your choice:4
Contents of the linked list are:45
12
87
```

Practical:21

AIM: Write a program to implement following operations on the circular linked list. Delete a first node.

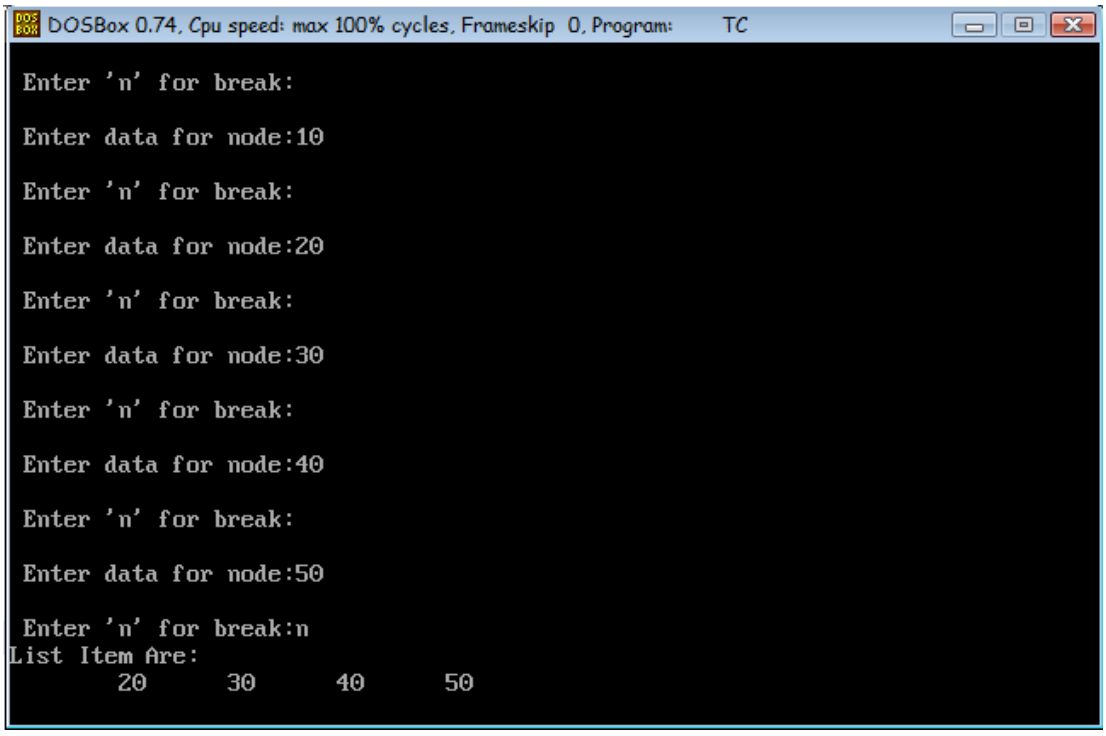
```
        #include<stdio.h>
        #include<conio.h>
        #include<stdlib.h>
struct link
{       int data;
    struct link *next;
};
         int i=0;

    struct link *node,*start,*ptr,*new1;
        void create_link(struct link *node)
    {   char ch;
        start->next=NULL;
        node=start;
        fflush(stdin);
        printf("\n Enter 'n' for break:");
        ch=getchar();
        while(ch!='n')
{
node->next=(struct link *)malloc(sizeof(struct link));
node=node->next;
printf("\n Enter data for node:");
scanf("%d",&node->data);
node->next=start;
fflush(stdin);
printf("\n Enter 'n' for break:");
ch=getchar();
        i++;
        }
         }

void delete_first(struct link *node)
{
        node=start->next;
        ptr=start;
        if(i==0)
        {
                        printf("\n List is empty");             exit(0);
    1.
```

```
            }
          ptr->next=node->next;
          free(node);
          i-          }
void display(struct link *node)
{
          int count;
          node=start->next;
          count=i;
         while(count)
          {
          printf("\t%d",node->data);
          node=node->next;
          count--;    }
}
void main()
{
char ch;
clrscr();
create_link(node);
delete_first(node);
printf("List Item Are:\n");
 display(node);
      getch();             }
```

Output



Practical:22

AIM: Write a program to implement following operations on the circular linked list. Delete a node after specified position.

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

/* structure for a node */
struct Node
{
   int data;
   struct Node *next;
};

/* Function to insert a node at the beginning of
  a Circular linked list */
void push(struct Node **head_ref, int data)
{
   // Create a new node and make head as next
   // of it.
   struct Node *ptr1 =
      (struct Node *)malloc(sizeof(struct Node));
   ptr1->data = data;
   ptr1->next = *head_ref;

   /* If linked list is not NULL then set the
     next of last node */
   if (*head_ref != NULL)
   {
      // Find the node before head and update
      // next of it.
      struct Node *temp = *head_ref;
      while (temp->next != *head_ref)
         temp = temp->next;
      temp->next = ptr1;
   }
   else
      ptr1->next = ptr1; /*For the first node */

   *head_ref = ptr1;
}

/* Function to print nodes in a given
 circular linked list */
```

```
void printList(struct Node *head)
{
   struct Node *temp = head;
   if (head != NULL)
   {
      do
      {
         printf("%d ", temp->data);
         temp = temp->next;
      }
      while (temp != head);
   }

   printf("\n");
}

/* Function to delete a given node from the list */
void deleteNode(struct Node *head, int key)
{
   if (head == NULL)
      return;

   // Find the required node
   struct Node *curr = head, *prev;
   while (curr->data != key)
   {
      if (curr->next == head)
      {
         printf("\nGiven node is not found"
               " in the list!!!");
         break;
      }

      prev = curr;
      curr = curr -> next;
   }

   // Check if node is only node
   if (curr->next == head)
   {
      head = NULL;
      free(curr);
      return;
   }

   // If more than one node, check if
```

```
        // it is first node
    if (curr == head)
    {
       prev = head;
       while (prev -> next != head)
          prev = prev -> next;
       head = curr->next;
       prev->next = head;
       free(curr);
    }

    // check if node is last node
    else if (curr -> next == head)
    {
       prev->next = head;
       free(curr);
    }
    else
    {
       prev->next = curr->next;
       free(curr);
    }
}

/* Driver program to test above functions */
int main()
{
    /* Initialize lists as empty */
    struct Node *head = NULL;

    /* Created linked list will be 2->5->7->8->10 */
    push(&head, 2);
    push(&head, 5);
    push(&head, 7);
    push(&head, 8);
    push(&head, 10);

    printf("List Before Deletion: ");
    printList(head);

    deleteNode(head, 7);

    printf("List After Deletion: ");
    printList(head);

    return 0;
```

```
}
```

Output:

List Before Deletion: 10 8 7 5 2
List After Deletion: 10 8 5 2

Practical:23

AIM: Write a program which create binary search tree.

```
#include<stdio.h>
#include<alloc.h>
#include<conio.h>
#include<stdio.h>
struct tree {
        int info;
        struct tree *left;
        struct tree *right;
};
struct tree *insert(struct tree *,int);
void display(struct tree *);
//void postorder(struct tree *);
//void preorder(struct tree *);
//struct tree *delet(struct tree *,int);
//struct tree *search(struct tree *);
int main(void) {
        struct tree *root;
        int choice, item,item_no;
        root = NULL;
        clrscr();
        /* rear  = NULL;*/
        do {
                do {
                        printf("\n \t 1. Insert in Binary Tree  ");
                    //  printf("\n\t 2. Delete from Binary Tree ");
                        printf("\n\t 3. display of Binary tree");
                     //  printf("\n\t 4. Postorder traversal of Binary tree");
                    //   printf("\n\t 5. Preorder traversal of Binary tree");
                    //   printf("\n\t 6. Search and replace ");
                        printf("\n\t 7. Exit ");
                        printf("\n\t Enter choice : ");
                        scanf(" %d",&choice);
                        if(choice<1 || choice>7)
                            printf("\n Invalid choice - try again");
                }
                while (choice<1 || choice>7);
                switch(choice) {
                        case 1:
                                printf("\n Enter new element: ");
                        scanf("%d", &item);
                        root= insert(root,item);
                        printf("\n root is %d",root->info);
```

```
                        printf("\n Inorder traversal of binary tree is : ");
                    display(root);
                    break;

                    case 3:
                            printf("\n Inorder traversal of binary tree is : ");
                    display(root);
                    break;
                //  case 4:
                  //            printf("\n Postorder traversal of binary tree is : ");
                   //  postorder(root);
                  //  break;
                    //case 5:
                    //          printf("\n Preorder traversal of binary tree is : ");
//  //                  preorder(root);
     //                 break;
  //
                    default:
                             printf("\n End of program ");
                }
                /* end of switch */
        }
        while(choice !=7);
        return(0);
}
struct tree *insert(struct tree *root, int x) {
        if(!root) {
                root=(struct tree*)malloc(sizeof(struct tree));
                root->info = x;
                root->left = NULL;
                root->right = NULL;
                return(root);
        }
        if(root->info > x)
            root->left = insert(root->left,x); else {
                if(root->info < x)
                        root->right = insert(root->right,x);
        }
        return(root);
}
void display(struct tree *root) {
        if(root != NULL) {
                display(root->left);
                printf(" %d",root->info);
                display(root->right);
        }
```

```
                return;
}
/*void postorder(struct tree *root) {
        if(root != NULL) {
                postorder(root->left);
                postorder(root->right);
                printf(" %d",root->info);
        }
        return;
}
void preorder(struct tree *root) {
        if(root != NULL) {
                printf(" %d",root->info);
                preorder(root->left);
                preorder(root->right);
        }
        return;
}
```

Practical:24

AIM: Implement recursive or non-recursive tree traversing methods inorder traversal.

Step 1: Start the process.

Step 2: Initialize and declare variables.

Step 3: Enter the choice. Inorder / Preorder / Postorder.

Step 4: If choice is Inorder then

- Traverse the left subtree in inorder.
- Process the root node.
- Traverse the right subtree in inorder.

Step 5: If choice is Preorder then

- Process the root node.
- Traverse the left subtree in preorder.
- Traverse the right subtree in preorder.

Step 6: If choice is postorder then

- Traverse the left subtree in postorder.
- Traverse the right subtree in postorder.
- Process the root node.

Step7: Print the Inorder / Preorder / Postorder traversal.

Step 8: Stop the process.

PROGRAM

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

typedef struct treenode

        {
         int data;
         struct treenode *left;
         struct treenode *right;
        }tnode;

tnode *insertion(int,tnode*);
void inorder(tnode *);
```

```
void main()
{
 tnode *T=NULL;
 int ch1,n;
 char ch2;
                        printf("\n\nenter the element to be inserted   :");
                        scanf("%d",&n);
                        T=insertion(n,T);


                        inorder(T);

 getch();
}
tnode *insertion(int x,tnode *T)
{
 if(T==NULL)
          {
           T=(tnode *)malloc(sizeof(tnode));
           if(T==NULL)
                    printf("\nout of space");
           else
                   {
                    T->data=x;
                    T->left=T->right=NULL;
                   }    } else
          {
           if(x<(T->data))
                    T->left=insertion(x,T->left);
           else
                   {
                    if(x>T->data)
                             T->right=insertion(x,T->right);
                   }   }
 return T;
}

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

Practical:25

AIM: Implement recursive and non-recursive tree traversing method of preorder traversal.

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

typedef struct treenode

        {
         int data;
         struct treenode *left;
         struct treenode *right;
        }tnode;

tnode *insertion(int,tnode*);
void preorder(tnode *);
void inorder(tnode *);
void postorder(tnode *);

void main()
{
 tnode *T=NULL;
 int ch1,n;
 char ch2;
                        printf("\n\nenter the element to be inserted   :");
                        scanf("%d",&n);
                        T=insertion(n,T);

                :
                        inorder(T);

                        preorder(T);




 getch();
}


tnode *insertion(int x,tnode *T)
{
```

```
    if(T==NULL)
        {
        T=(tnode *)malloc(sizeof(tnode));
        if(T==NULL)
                printf("\nout of space");
        else
                {
                T->data=x;
                T->left=T->right=NULL;
                }
        }
 else
        {
        if(x<(T->data))
                T->left=insertion(x,T->left);
        else
                {
                if(x>T->data)
                        T->right=insertion(x,T->right);
                }
        }
 return T;
}

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

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

Practical:26

AIM: Write a program to implement Merge Sort.

PROGRAM:

```
#include<stdio.h>
#define MAX 50

void mergeSort(int arr[],int low,int mid,int high);
void partition(int arr[],int low,int high);

int main(){

   int merge[MAX],i,n;

   printf("Enter the total number of elements: ");
   scanf("%d",&n);

   printf("Enter the elements which to be sort: ");
   for(i=0;i<n;i++){
      scanf("%d",&merge[i]);
   }

   partition(merge,0,n-1);

   printf("After merge sorting elements are: ");
   for(i=0;i<n;i++){
      printf("%d ",merge[i]);
   }

  return 0;
}

void partition(int arr[],int low,int high){

   int mid;

   if(low<high){
      mid=(low+high)/2;
      partition(arr,low,mid);
      partition(arr,mid+1,high);
      mergeSort(arr,low,mid,high);
   }
}

void mergeSort(int arr[],int low,int mid,int high){

   int i,m,k,l,temp[MAX];
```

```
    l=low;
    i=low;
    m=mid+1;

    while((l<=mid)&&(m<=high)){

        if(arr[l]<=arr[m]){
            temp[i]=arr[l];
            l++;
        }
        else{
            temp[i]=arr[m];
            m++;
        }
        i++;
    }

    if(l>mid){
        for(k=m;k<=high;k++){
            temp[i]=arr[k];
            i++;
        }
    }
    else{
        for(k=l;k<=mid;k++){
            temp[i]=arr[k];
            i++;
        }
    }

    for(k=low;k<=high;k++){
        arr[k]=temp[k];
    }
}
```

Sample output:

Enter the total number of elements: 5
Enter the elements which to be sort: 2 5 0 9 1
After merge sorting elements are: 0 1 2 5 9

Practical:27

AIM: Write a program to implement Bubble Sort.

PROGRAM:

```
#include<stdio.h>
void main()
{  int s,temp,i,j,a[20];
  clrscr();
  printf("Enter total numbers of elements: ");
  scanf("%d",&s);
 printf("Enter %d elements: ",s);
  for(i=0;i<s;i++)
     scanf("%d",&a[i]);
  //Bubble sorting algorithm
  for(i=0;i<s;i++)
 {
     for(j=i;j<s;j++)
    {
             if(a[j]>a[j+1])
            {
                temp=a[j];
               a[j]=a[j+1];
               a[j+1]=temp;
             } }
  }

  printf("After sorting: ");
  for(i=0;i<s;i++)
     printf(" %d",a[i]);
getch();
}
```

Result:

```
Enter total numbers of elements: 5
Enter 5 elements: 6
4
2
8
9
After sorting:  2 4 6 8 9
```

Practical:28

AIM: Write a program to implement Binary Search.

OBJECTIVE:

PROGRAM:

```
#include <stdio.h>
#include<conio.h>
int main()
{
  int c, first, last, middle, n, search, array[100];
 clrscr();
  printf("Enter number of elements\n");
  scanf("%d",&n);

  printf("Enter %d integers\n", n);

  for ( c = 0 ; c < n ; c++ )
    scanf("%d",&array[c]);

  printf("Enter value to find\n");
  scanf("%d",&search);

  first = 0;
  last = n - 1;
  middle = (first+last)/2;

  while( first <= last )
  {
    if ( array[middle] < search )
      first = middle + 1;
    else if ( array[middle] == search )
    {
      printf("%d found at location %d.\n", search, middle+1);
      break;
    }
    else
      last = middle - 1;

    middle = (first + last)/2;
  }
  if ( first > last )
    printf("Not found! %d is not present in the list.\n", search);
return 0;
```

Result:

```
Enter number of elements
7
Enter 7 integers
2
33
45
65
68
78
89
Enter value to find
68
68 found at location 5.
```

THE END

6. IMPLEMENT CIRCULAR QUEUE

AIM:- Write a program to implement Insert, Delete and Display on Circular queue.

OBJECTIVES:

1. Start the program
2. To insert an element,

   Step-i: If "rear" of the queue is pointing to the last position then go to step-ii or else step-iii

   Step-ii: make the "rear" value as 0

   Step-iii: increment the "rear" value by one

   Step-iv: a. if the "front" points where "rear" is pointing and the queue holds a not NULL value for it, then its a "queue overflow" state, so quit; else go to step-b

   b. insert the new value for the queue position pointed by the "rear"

3. To delete the particular item from circular queue

   Step-i: If the queue is empty then say "empty queue" and quit; else continue

   Step-ii: Delete the "front" element

   Step-iii: If the "front" is pointing to the last position of the queue then step-iv else step-v

   Step-iv: Make the "front" point to the first position in the queue and quit

   Step-v: Increment the "front" position by one

4. Terminate the program.

PROGRAM:-

```
#include <stdio.h>
#include<ctype.h>
# define MAXSIZE 200

int cq[MAXSIZE];
int front,rear;

void main()
{
void add(int,int [],int,int,int);
int del(int [],int ,int ,int );
int will=1,i,num;
front = 1;
rear = 1;

clrscr();
printf("Program for Circular Queue demonstration through array");
while(will ==1)
{
printf("MAIN MENU:
        1.Add element to Circular Queue
        2.Delete element from the Circular Queue
");
scanf("%d",&will);

switch(will)
{
case 1:
        printf("Enter the data... ");
        scanf("%d",&num);
        add(num,cq,MAXSIZE,front,rear);
        break;
case 2: i=del(cq,MAXSIZE,front,rear);
        printf("Value returned from delete function is  %d ",i);
        break;
default: printf("Invalid Choice . ");
}

printf(" Do you want to do more operations on Circular Queue ( 1 for yes, any other key
to exit)
                ");
scanf("%d" , &will);
} //end of  outer while
}           //end of main
```

```
void add(int item,int q[],int MAX,int front,int rear)
{
rear++;
rear= (rear%MAX);
if(front ==rear)
        {
        printf("CIRCULAR QUEUE FULL");
        return;
        }
else
        {
        cq[rear]=item;
        printf("Rear = %d    Front = %d ",rear,front);
        }
}
int del(int q[],int MAX,int front,int rear)
{
int a;
if(front == rear)
        {
        printf("CIRCULAR STACK EMPTY");
        return (0);
        }
else
        {
        front++;
        front = front%MAX;
        a=cq[front];
        return(a);
        printf("Rear = %d    Front = %d ",rear,front);
        }
}
```

RESULT:-

MAIN MENU:

1.Add element to Circular Queue
2.Delete element from the Circular Queue

Enter the data

56, 44,66,…….

The given program is implemented, executed, tested and verified successfully.

7. IMPLEMENTATION OF DEQUEUE

AIM:- To implement Dequeue in C.

OBJECTIVES:

Take two variants double ended queue.

- Input resrticted dqueue.
- Output restricted dqueue.

PROGRAM:-

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

 # define MAX 5

 int deque_arr[MAX];
 int left = -1;
 int right = -1;

 main()
 {
 int choice;

 printf("1.Input restricted dequeue\n");
 printf("2.Output restricted dequeue\n");
 printf("Enter your choice : ");
 scanf("%d",&choice);

 switch(choice)
 {
 case 1 :
 input_que();
 break;
 case 2:
 output_que();
 break;
 default:
 printf("Wrong choice\n");
 }/*End of switch*/
```

```
}/*End of main()*/

input_que()
{
int choice;
while(1)
{
printf("1.Insert at right\n");
printf("2.Delete from left\n");
printf("3.Delete from right\n");
printf("4.Display\n");
printf("5.Quit\n");
printf("Enter your choice : ");
scanf("%d",&choice);

switch(choice)
{
case 1:
insert_right();
break;
case 2:
delete_left();
break;
case 3:
delete_right();
break;
case 4:
display_queue();
break;
case 5:
exit();
default:
printf("Wrong choice\n");
}/*End of switch*/
}/*End of while*/
}/*End of input_que() */

output_que()
{
int choice;
while(1)
{
printf("1.Insert at right\n");
printf("2.Insert at left\n");
printf("3.Delete from left\n");
printf("4.Display\n");
```

```
    printf("5.Quit\n");
printf("Enter your choice : ");
scanf("%d",&choice);

switch(choice)
{
case 1:
insert_right();
break;
case 2:
insert_left();
break;
case 3:
delete_left();
break;
case 4:
display_queue();
break;
case 5:
exit();
default:
printf("Wrong choice\n");
}/*End of switch*/
}/*End of while*/
}/*End of output_que() */

insert_right()
{
int added_item;
if((left == 0 && right == MAX-1) || (left == right+1))
{
printf("Queue Overflow\n");
return;
}
if (left == -1) /* if queue is initially empty */
{
left = 0;
right = 0;
}
else
if(right == MAX-1) /*right is at last position of queue */
right = 0;
else
right = right+1;
printf("Input the element for adding in queue : ");
scanf("%d", &added_item);
```

```
    deque_arr[right] = added_item ;
}/*End of insert_right()*/

insert_left()
{
int added_item;
if((left == 0 && right == MAX-1) || (left == right+1))
{
printf("Queue Overflow \n");
return;
}
if (left == -1)/*If queue is initially empty*/
{
left = 0;
right = 0;
}
else
if(left== 0)
left=MAX-1;
else
left=left-1;
printf("Input the element for adding in queue : ");
scanf("%d", &added_item);
deque_arr[left] = added_item ;
}/*End of insert_left()*/

delete_left()
{
if (left == -1)
{
printf("Queue Underflow\n");
return ;
}
printf("Element deleted from queue is : %d\n",deque_arr[left]);
if(left == right) /*Queue has only one element */
{
left = -1;
right=-1;
}
else
if(left == MAX-1)
left = 0;
else
left = left+1;
}/*End of delete_left()*/
```

```
delete_right()
{
if (left == -1)
{
printf("Queue Underflow\n");
return ;
}
printf("Element deleted from queue is : %d\n",deque_arr[right]);
if(left == right) /*queue has only one element*/
{
left = -1;
right=-1;
}
else
if(right == 0)
right=MAX-1;
else
right=right-1;
}/*End of delete_right() */

display_queue()
{
int front_pos = left,rear_pos = right;
if(left == -1)
{
printf("Queue is empty\n");
return;
}
printf("Queue elements :\n");
if( front_pos <= rear_pos )
{
while(front_pos <= rear_pos)
{
printf("%d ",deque_arr[front_pos]);
front_pos++;
}
}
else
{
while(front_pos <= MAX-1)
{
printf("%d ",deque_arr[front_pos]);
front_pos++;
}
front_pos = 0;
while(front_pos <= rear_pos)
```

```
  {
printf("%d ",deque_arr[front_pos]);
front_pos++;
}
}/*End of else */
printf("\n");
}/*End of display_queue() */
```

RESULT::

```
1.Insert at right
2.Insert at left
3.Delete from left
4.Display
5.Quit
Enter your choice : 1
Input the element for adding in queue : 55
```

The given program is implemented, executed, tested and verified successfully

1. SINGLY LINKED LIST

AIM:- Write a program to implement Insert, Delete and Display on Singly Linked List

OBJECTIVES:-

1. Start the program.
2. Get the choice from the user.
3. If the choice is to add records, get the data from the user and add them to the list.
4. If the choice is to delete records, get the data to be deleted and delete it from the list.
5. If the choice is to display number of records, count the items in the list and display.
6. If the choice is to search for an item, get the item to be searched and respond yes if the item is found, otherwise no.
7. Terminate the program

PROGRAM:-

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#define NULL 0
struct info
{
       int data;
       struct info *next;
};
struct info *head,*temp,*disp;
void additem();
void delitem();
void display();
int size();
void search();
void main()
{
      int choice;
```

```
        clrscr();
    while(1)
    {
        printf("\n1.Add records");
        printf("\n2.Delete records");
        printf("\n3.Display records");
        printf("\n4.Count no. of items in the list");
        printf("\n5.Searching an item in the list");
        printf("\n6.Exit");
        printf("\nEnter your choice:");
        scanf("%d",&choice);
        fflush(stdin);
        switch(choice)
        {
            case 1:
              additem();
              break;
            case 2:
              delitem();
              break;
            case 3:
              display();
              break;
            case 4:
              printf("\nThe size of the list is %d",size());
              break;
            case 5:
              search();
              break;
            case 6:
              exit(0);
        }
    }
}
void additem()
{
    struct info *add;
    char proceed='y';
    while(toupper(proceed)=='Y')
    {
        add=(struct info*)malloc(sizeof(struct info));
        printf("Enter data:");
        scanf("%d",&add->data);
        fflush(stdin);
        if(head==NULL)
        {
```

```
                    head=add;
                add->next=NULL;
                temp=add;
            }
            else
            {
                temp->next=add;
                add->next=NULL;
                temp=add;
            }
            printf("\nWant to proceed y/n");
            proceed=getchar();
            fflush(stdin);
        }
}
void delitem()
{
        struct info *curr,*prev;
        int tdata;
        if(head==NULL)
        {
            printf("\nNo records to delete");
            return;
        }
        printf("\nEnter the data to delete");
        scanf("%d",&tdata);
        fflush(stdin);
        prev=curr=head;
        while((curr!=NULL)&&(curr->data!=tdata))
        {
            prev=curr;
            curr=curr->next;
        }
        if(curr==NULL)
        {
            printf("\nData not found");
            return;
        }
        if(curr==head)
            head=head->next;
        else
        {
            /*for inbetween element deletion*/
            prev->next=curr->next;
            /*for the last element deletion*/
            if(curr->next==NULL)
```

```
                temp=prev;
        }
        free(curr);
}
void display()
{
        if(head==NULL)
        {
                printf("\nNo data to display");
                return;
        }
        for(disp=head;disp!=NULL;disp=disp->next)
        {
                printf("Data->%d",disp->data);
        }
}
int size()
{
        int count=0;
        if(head==NULL)
             return count;
        for(disp=head;disp!=NULL;disp=disp->next)
             count++;
        return count;
}
void search()
{
        int titem,found=0;
        if(head==NULL)
        {
                printf("\nNo data in the list");
                return;
        }
        printf("\Enter the no. to search:");
        scanf("%d",&titem);
        for(disp=head;disp!=NULL&&found==0;disp=disp->next)
        {
                if(disp->data==titem)
                   found=1;
        }
        if(found==0)
                printf("\nSearch no. is not present in the list");
        else
                printf("\nSearch no. is present in the list");
        return;
}
```

RESULT:-

```
1.Add records
2.Delete records
3.Display records
4.Count no. of items in the list
5.Searching an item in the list
6.Exit
Enter your choice:1
Enter data:12
Want to proceed y/ny
Enter data:13
Want to proceed y/ny
Enter data:41
Want to proceed y/nn

1.Add records
2.Delete records
3.Display records
4.Count no. of items in the list
5.Searching an item in the list
6.Exit
Enter your choice:3
Data->12Data->13Data->41
1.Add records
2.Delete records
3.Display records
4.Count no. of items in the list
5.Searching an item in the list
6.Exit
Enter your choice:4
The size of the list is 3
1.Add records
2.Delete records
3.Display records
4.Count no. of items in the list
5.Searching an item in the list
6.Exit
Enter your choice:2
Enter the data to delete13

1.Add records
2.Delete records
3.Display records
4.Count no. of items in the list
```

5.Searching an item in the list
6.Exit
Enter your choice:3
Data->12Data->41
1.Add records
2.Delete records
3.Display records
4.Count no. of items in the list
5.Searching an item in the list
6.Exit
Enter your choice:5
Enter the no. to search:13
Search no. is not present in the list
1.Add records
2.Delete records
3.Display records
4.Count no. of items in the list
5.Searching an item in the list
6.Exit
Enter your choice:6

9. DOUBLY LINKED LIST

AIM : To write a program to implement doubly linked list using linked list.

OBJECTIVES:

Step 1: Declare header and pointer variables
Step 2: Display the choices
Step 3: If choice is 1 the get the element to be inserted in beginning and call ins_beg function.
Step 4: If choice is 2 the get the element to be inserted in the end and call the ins_end function
Step 5: If choice is 3 then get the element to be deleted and call deletion function.
Step 6: If choice is 4 then call display duncation
Step 7: If choice is default the exit the program
Step 8: Terminate the program execution.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void display(struct node *first);
struct node
```

```
        {
        int data;
        struct node *lptr,*rptr;
}*head;
struct node *ins_beg(int x,struct node *first)
{
        struct node *new1,*cur,*prev;
        new1=malloc(sizeof(struct node));
        if(first==NULL)
        {
                new1->data=x;
                new1->lptr=NULL;
                new1->rptr=NULL;
                return new1;
        }
        else
        {
                new1->data=x;
                new1->lptr=NULL;
                new1->rptr=first;
                return new1;
        }
}
struct node *ins_end(int x,struct node *first)
{
        struct node *new1,*cur,*prev;
        new1=malloc(sizeof(struct node));
        if(first==NULL)
        {
                new1->data=x;
                new1->lptr=NULL;
                new1->rptr=NULL;
                return new1;
        }
        else
        {
                cur=first;
                while(cur->rptr!=NULL)
                {
                        prev=cur;
                        cur=cur->rptr;
                }
                cur->rptr=new1;
                new1->data=x;
                new1->lptr=cur;
                new1->rptr=NULL;
```

```
                return first;
        }
}
struct node *deletion(struct node *first,int del)
{
        struct node *prev,*cur;
        cur=first;
        if(first==NULL)
        {
                printf("\n\nNo data present!!!");
                getch();
        }
        else if(first->data==del)
        {
                printf("\n\nData %d is deleted",first->data);
                first=first->rptr;
                getch();
                return first;
        }
        else
        {
                while(cur->rptr!=NULL && cur->data!=del)
                {
                        prev=cur;
                        cur=cur->rptr;
                }
                if(cur->rptr==NULL && cur->data!=del)
                        printf("\n\nData is not present!!!");
                else if(cur->rptr!=NULL && cur->data==del)
                {
                        prev->rptr=cur->rptr;
                        (cur->rptr)->lptr=prev;
                        printf("\n\nData % d is deleted",cur->data);
                }
                else if(cur->rptr==NULL && cur->data==del)
                {
                        prev->rptr=NULL;
                        printf("\n\nData %d is deleted:",cur->data);
                }
                getch();
                return first;
        }
}
void main()
{
        int x,ch,del;
```

```
                head=NULL;
        clrscr();
        printf("\n1.Insert in Begining\n2.Insert in the End\n3.Delete\n4.Display");
        while(1)
        {
                printf("\n\nEnter your Choice :: ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                                printf("\n\nEnter the element to be inserted::");
                                scanf("%d",&x);
                                head=ins_beg(x,head);
                                break;
                        case 2:
                                printf("\n\nEnter the element to be inserted::");
                                scanf("%d",&x);
                                head=ins_end(x,head);
                                break;
                        case 3:
                                printf("\n\nEnter the element to be deleted::");
                                scanf("%d",&del);
                                head=deletion(head,del);
                                break;
                        case 4:
                                display(head);
                                break;
                        default:
                                printf("\n\nInvalid Choice.....");
                                getch();
                                exit(0);
                }
        }
}
void display(struct node *first)
{
        struct node *temp;
        temp=first;
        if(temp==NULL)
                printf("\n\nList is empty!!!");
        while(temp!=NULL)
        {
                printf("%d ->",temp->data);
                temp=temp->rptr;
        }
        getch();
```

}

RESULT:-

1.Insert in Begining
2.Insert in the End
3.Delete
4.Display

Enter your Choice :: 1

Enter the element to be inserted::2

Enter your Choice :: 1

Enter the element to be inserted::3

Enter your Choice :: 4
3 ->2 ->

Enter your Choice :: 2

Enter the element to be inserted::1

Enter your Choice :: 2

Enter the element to be inserted::5

Enter your Choice :: 4
3 ->2 ->1 ->5 ->

Enter your Choice :: 3
Enter the element to be deleted::1
Data 1 is deleted
Enter your Choice :: 4
3 ->2 ->5 ->

10. IMPLEMENTATION OF TREE TRAVERSALS

AIM:- To write a 'C' program to implement an expression tree. Produce its pre-order, in-order, and post-order traversals.

OBJECTIVES:

Step 1: Start the process.

Step 2: Initialize and declare variables.

Step 3: Enter the choice. Inorder / Preorder / Postorder.

Step 4: If choice is Inorder then

- Traverse the left subtree in inorder.
- Process the root node.
- Traverse the right subtree in inorder.

Step 5: If choice is Preorder then

- Process the root node.
- Traverse the left subtree in preorder.

- Traverse the right subtree in preorder.

Step 6: If choice is postorder then

- Traverse the left subtree in postorder.
- Traverse the right subtree in postorder.
- Process the root node.

Step7: Print the Inorder / Preorder / Postorder traversal.

Step 8: Stop the process.

PROGRAM

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

typedef struct treenode

        {
         int data;
         struct treenode *left;
         struct treenode *right;
        }tnode;

tnode *insertion(int,tnode*);
void preorder(tnode *);
void inorder(tnode *);
void postorder(tnode *);

void main()
{
 tnode *T=NULL;
 int ch1,n;
 char ch2;
 do
        {
         clrscr();
         printf("\n\t\t****Operation With Tree****");
         printf("\n\t1.Insertion");
         printf("\n\t2.Inorder Traversal");
         printf("\n\t3.Preorder Traversal");
         printf("\n\t4.Postorder Traversal");
         printf("\n\tEnter Your Choice      :");
         scanf("%d",&ch1);
         switch(ch1)
                {
                 case 1:
```

```
                    printf("\n\nenter the element to be inserted   :");
                scanf("%d",&n);
                T=insertion(n,T);
                break;
            case 2:
                inorder(T);
                break;
            case 3:
                preorder(T);


                break;
            case 4:
                postorder(T);
                break;
            default:
                printf("\n\nInvalid Option");
                break;
            }
        printf("\n\nDo you want to continue y/n     : ");
        scanf("%s",&ch2);
        }while(ch2=='y');
 getch();
}


tnode *insertion(int x,tnode *T)
{
 if(T==NULL)
        {
        T=(tnode *)malloc(sizeof(tnode));
        if(T==NULL)
            printf("\nout of space");
        else
            {
            T->data=x;
            T->left=T->right=NULL;
            }
        }
 else
        {
        if(x<(T->data))
            T->left=insertion(x,T->left);
        else
            {
            if(x>T->data)
```

```
                    T->right=insertion(x,T->right);
                }
        }
 return T;
}
```

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

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

RESULT:

Insertion : 4 , 6, 2, 3, 1, 88
Preorder Traversal :: 4, 2, 1, 3, 6, 88

Postorder Traversal :: 1, 3, 2, 88, 6, 4
Inorder Traversal :: 1, 2, 3, 4. 6, 88