Gropu A

1.Perform following String operations with and without pointers to arrays (without using the library functions):

a. substring

b. palindrome

c. compare

d. copy

e. reverse

2.Implement Database Management using array of structures with operations Create, Display, Modify, Append, Search and Sort. (For any database like Employee or Bank database with and without pointers to structures)

3. Implement Stack and Queue using arrays.

4. Create a singly linked list with options:

a. Insert (at front, at end, in the middle)

b. Delete (at front, at end, in the middle)

c. Display

d. Display Reverse

e. Revert the SLL

5. Implement Binary search tree with operations Create, search, and recursive traversal.

6. Implement Graph using adjacency Matrix with BFS & DFS traversal.

Group B-

7. Implement stack and queue using linked list.

8. Add two polynomials using linked list.

9. Evaluate postfix expression (input will be postfix expression).

Group C-

10. Implement Circular Linked List with various operations

//Experiment-1_part_A
String operations without pointers

```
//Roll No:        Class:          Div:            Name:

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

#define MAX 20

void main()
{
int choice,temp,len,pos,temp2,l,r,flag=0;
char stri[MAX],stro[MAX];
clrscr();

printf("\nEnter a string with max character %d:=>",MAX);
scanf("%s",stri);

for(temp=0,len=0;stri[temp]!='\0';temp++,len++);
printf("\nLength of the given String is:=>%d",len);

while(1)
{
printf("\n****Menu****\n1.Print substring\n2.Copy string into other array\n
3.Reverse the given string\n4.Check string as palindrome\n5.Compare
string\n6.Exit\n");

scanf("%d",&choice);

switch(choice)
{
case 1:
printf("\nEnter the position from where the sub string must get displayed:=>");
scanf("%d",&pos);
for(temp=0;temp<len-pos;temp++)
{
stro[temp]=stri[pos+temp];
}
stro[temp]='\0';
printf("\nSubString is:%s",stro);
break;

case 2:
for(temp=0;stri[temp]!='\0';temp++)
{
stro[temp]=stri[temp];
}
stro[temp]='\0';
printf("\nEntered string is:=>%s",stri);
printf("\nString after copying:=>%s",stro);
break;

case 3:
printf("\nentered string is:=>%s",stri);
for(temp=len-1,temp2=0;temp>=0;temp--,temp2++)
{
stro[temp2]=stri[temp];
}
stro[temp2]='\0';
printf("\nString after reversing:=>%s",stro);
break;

case 4:
printf("\nEntered string is:=>%s",stri);
l=0,r=len-1;
while(stri[l++]!=stri[r--])
{
flag=1;
break;
}
if(flag==1)
printf("\nEntered string is not a plaindrome");
else
printf("\nEntered string is a palindrome");
break;

case 5:
printf("\nEnter another string to perform comparision:=>");
scanf("%s",stro);
temp=0;
while(stri[temp]==stro[temp] && stri[temp]!='\0')
```

//Experiment-1_Part-B
String operations with pointers

```
//Roll No:        Class:          Div:            Name:
#include<conio.h>
#include <stdio.h>
#include <stdlib.h>

#define MAX 20
#define pf printf
#define sf scanf

void substring(char *src, char *dest, int len);
void strcopy(char *src, char *dest);
void reverse(char *src,char *dest, int len);
int palindrome(char *src, char *dest, int len);
int compare(char *src, char *dest, int len);

int main()
{
char source[MAX],dest[MAX];
int choice,len,flag;
clrscr();
pf("\nEnter a string with max characters %d",MAX);
sf("%s",source);
for(len=0;source[len]!='\0';len++);
pf("\nLength of the given string is :=>%d\n",len);

while(1)
{
        pf("\n****MENU****\n");
        pf("\n1.String copy\n2.Print substing from main string");
        pf("\n3.Display reverse string\n4.check is entered string is palindrome");
        pf("\n5.Compare string\n6.Exit:=>");
        sf("%d",&choice);
        switch(choice)
        {
                case 1:strcopy(source,dest);
                pf("\nCopied string is: %s",dest);
                break;

                case 2: substring(source,dest,len);
                pf("\nSub string is:=>%s",dest);
                break;

                case 3: reverse(&source[len-1],dest,len);
                pf("\nSub string is:=>%s",dest);
                break;

                case 4: flag=palindrome(source,&source[len-1],len);
                if(flag==1)
                pf("\nEntered string is palindrome..");
                else
                pf("\nEntered string is not a palindrome..");
                break;

                case 5 :
                pf("\nEnter string to comapre:=>");
                sf("%s",dest);
                flag=compare(source,dest,len);
                if(flag==1)
                {
                pf("\nBoth strings are identical");
                }
                else
                {
                pf("\nBoth strings are not identical");
                }
                break;

                case 6: exit(1);
        }
}
        return(0);
}
void strcopy(char *source,char *dest)
{
        int temp;
        for(temp=0;*source!='\0';temp++)
        {
                *dest = *source;
```

```
{
if(stri[temp]>stro[temp])
printf("\nstri > stro");
else if(stri[temp]<stro[temp])
printf("\nstro > stri");
else
printf("stri = stro");
temp++;}
break;

case 6:
exit(1);

}//switch_ends
}//while_ends
}//main_ends
```

```
                dest++;
                source++;
        }
        *dest='\0';
}
void substring(char *source,char *dest,int len)
{
        int temp=0,pos;
        pf("\nEnter the position from which elememts need to be displayed:\n");
        sf("%d",&pos);
        for(temp=0;temp<pos;temp++)
        {
                source++;
        }
        for(temp=pos;temp<len;temp++)
        {
                *dest=*source;
                dest++;
                source++;
        }
        *dest='\0';
}
void reverse(char *source, char *dest,int len)
{
        int temp;
        for(temp=0;temp<len;temp++)
        {
                *dest=*source;
                dest++;
                source--;
        }
        *dest='\0';
}
int palindrome(char *source,char *dest, int len)
{
        int temp,flag=1;
        for(temp=0;temp<len/2;temp++)
        {
        if(*source!=*dest)
        {
        flag=0;
        break;
        }
}
return flag;
}
int compare(char *source,char *dest,int len)
{
        int temp;
        for(temp=0;temp<len;temp++)
        {
        if(*source++!=*dest++)
        {
        return 0;
        }
}
return 1;
}
```

//Experiment-2_part_A
Employee Database management without pointers

```
//Roll No:        Class:          Div:            Name:

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

//DECLARATION OF STRUCTURE
struct employee{
                int id;
                char name[20];
                int age;
                long int salary;
            }e[15];      //DECLARING ARRAYS OF STRUCTUREHAVING 15
ELEMENTS

void create(int p);
void display(int y);
void modify(int m,int c);
void append(int x);
int search(int key,int a);
void sort(int b);



void main()
{                       //DECLARING LOCAL VARIABLE FOR main() FUNCTION
    int choice,no,i,n,num;
    char op;
    clrscr();
    printf("\n enter how many records ?\n");
    scanf("%d",&n);
    create(n);              //calling create() function
    do
       { printf("\n menu\n 1.display\n 2.modify\n 3.append\n 4.search\n
5.sort\n");
         printf("\n enter your choice\n");
         scanf("%d",&choice);

switch(choice)          //switch structure
        {
          case 1:display(n);
                 break;
          case 2:printf("\nenter the employee id to be modify:-\n");
                 scanf("%d",&num);
                 modify(num,n);     //calling modify() function
                 break;
          case 3:append(n);
                 n++;
                 break;
          case 4:printf("\n enter the employee number to be searched:-\n");
                 scanf("%d",&num);
                 i=search(num,n);
                 //calling search() function which returns int value
                 if(i==-1)
                 printf("\n employee not found\n");
                 else
                 printf("\n employee found at %d location\n",i);
                 break;
        case 5:sort(n);         //calling sort function
                 break;
        default:printf("\n invalid choice.\n");
                  break;
        }

              printf("\ndo you wan to continue\n");
              op=getch();
              putch(op);

}while(op=='Y'||op=='y');

        getch();
    }
```

//Experiment-2_part_B
Employee Database management with pointers

```
//Roll No:        Class:          Div:            Name:

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

typedef struct employee
{
char name[20];
int empid,salary;
}emp;

emp e[10];

void create();
void display();
void modify();
void search();
void sort();
int n;

main()
{
int option;
//clrscr();
while(1)
{
printf("\nMenu:\n1.create\n2.display\n3.search\n4.sort\n5.modify\n6.exit");
scanf("%d",&option);
switch(option) //switch
{
case 1: create();
break;
case 2: display();
break;
case 3: search();
break;
case 4: sort();
break;
case 5: modify();
break;
case 6: exit(0);
} //end of switch
getch();
} //end main
}


void create() //start of create
{
int i;
emp *eptr=e;
printf("enter no of employee");
scanf("%d",&n);
printf("Enter empname\tempid\tsalary");
for(i=0;i<n;i++)
{
scanf("%s%d%d",eptr->name,&eptr->empid,&eptr->salary);
eptr++;
}
}//end of create


void display()//start of display
{
int i;
emp *eptr=e ;
printf("employee data name,id and salary");
for(i=0;i<n;i++)
{
printf("\n%s %d %d",eptr->name,eptr->empid,eptr->salary);
eptr++;
}
} //end of display


void search() //start of serach
{
int i,flag,keyid;
```

```
void create(int p)
  { int i;
    for(i=0;i<p;i++)
          { printf("\n enter information for [%d] employee -->\n",i+1);
            printf("\n enter the employee id :->");
            scanf("%d",&e[i].id);
            printf("\n enter name\n");
            scanf("%s",e[i].name);
            printf("\n enter age\n");
            scanf("%d",&e[i].age);
            printf("\n enter salary:->\n");
            scanf("%ld",&e[i].salary);
            }
    }


void display(int y)
    { int i;
      printf("\n the information for employees\n");
      printf("\n id\tname\t\tage\tsalary\n");
      for(i=0;i<y;i++)
      printf("\n %d\t%s\t\t%d\t%d",e[i].id,e[i].name,e[i].age,e[i].salary);
    }


void modify(int m,int c)
    { int pos;
      pos=search(m,c);
      if(pos==-1)
           printf("\n employee id is not existing.\n");
      else {
           printf("\n enter the information for [%d] employee again \n",pos);
           printf("\n enter the new employee id :-\n");
           scanf("%d",&e[pos].id);
           printf("\n enter new name\n");
           scanf("%s",e[pos].name);
           printf("\n enter new age\n");
           scanf("%d",&e[pos].age);
           printf("\n enter new salary:->\n");
           scanf("%ld",&e[pos].salary);
            }
    }


void append(int x)
  {     printf("\n enter the new record for [%d] employee \n",x+1);
           printf("\n enter the new employee id :-\n");
           scanf("%d",&e[x].id);
           printf("\n enter new name\n");
           scanf("%s",e[x].name);
           printf("\n enter new age\n");
           scanf("%d",&e[x].age);
           printf("\n enter new salary:->\n");
           scanf("%ld",&e[x].salary);
  }


int search(int key,int a)
  {
    int i;
    for (i=0;i<a;i++)
           if(key==e[i].id)
              return(i+1);

    return -1;

    }



void sort(int b)
    {
      int i,j;
      struct employee temp;
      for(i=0;i<b-1;i++)
           for(j=0;j<b-1-i;j++)
              if(e[j].id>e[j+1].id)
                    {  temp=e[j];
                       e[j]=e[j+1];
                       e[j+1]=temp;
                    }

      printf("\n records sorted in ascending order of ids.\n");

    }
```

```
emp *eptr=e ;
flag=0;
printf("enter keyid to be searched");
scanf("%d",&keyid);
for(i=0;i<n;i++)
{
if(keyid==eptr->empid)
{
flag++;
break;
}
eptr++;
}

if(flag==1)
printf("Record found at %d position",i+1);
else
printf("Record not found");
//end of search
}


void sort()
{
int i,j,temp,saltemp;
char nametemp[20];
emp *eptr=e ;
for(i=0;i<n-1;i++)
{
for(j=0;j<n-1-j;j++)
{
if(eptr->empid>(eptr+1)->empid)
{
temp=eptr->empid;
eptr->empid=(eptr+1)->empid;
(eptr+1)->empid=temp;
saltemp=eptr->salary;
eptr->salary=(eptr+1)->salary;
(eptr+1)->salary=saltemp;
strcpy(nametemp,eptr->name);
strcpy(eptr->name,(eptr+1)->name);
strcpy((eptr+1)->name,nametemp);
}
}
}
display();
}//end of sort*/


void modify()
{
int i,keyid;
emp *eptr=e;
printf("enter record to be modified");
scanf("%d",&keyid);
for(i=0;i<n;i++)
{
if(keyid==eptr->empid)
{
printf("enter name,id & salary for modification");
scanf("%s%d%d",eptr->name,&eptr->empid,&eptr->salary);
break;
}//end if
} //end of for
display();
}//end of modify
```

//Experiment-3-part_A
Stack using Array

```
//Roll No:        Class:          Div:          Name:
#include<stdio.h>
#include<stdlib.h>


#define MAX 10
int stack_arr[MAX];
int top = -1;
void push();
void pop();
int isEmpty();
int isFull();
void display();
main()
{
int choice,item;
clrscr();
while(1)
{
printf("\n1.Push\n");
printf("2.Pop\n");
printf("3.Display all stack elements\n");
printf("4.Quit\n");
printf("Enter your choice : ");
scanf("%d",&choice);
switch(choice)
{
case 1 :
push();
break;
case 2:
pop();
break;
case 3:
display();
break;
case 4:
exit(1);
default:
printf("\nWrong choice\n");
}/*End of switch*/
}/*End of while*/
}/*End of main()*/
void push()
{
int item;
if( isFull() )
{
printf("Stack Overflow\n");
}
else
{
printf("\nEnter the element:\n");
scanf("%d",&item);
top = top+1;
stack_arr[top] = item;
}
}/*End of push()*/
void pop()
{
int item;
if( isEmpty() )
{
printf("\nStack Underflow\n");
}
else
{
item = stack_arr[top];
printf("\nThe popped element : %d",item);
top = top-1;
}
}/*End of pop()*/
int isEmpty()
{
if( top == -1 )
return 1;
else
return 0;
}/*End of isEmpty*/
int isFull()
{
```

//Experiment-7_part_A
Stack using linklist

```
//Roll No:        Class:          Div:          Name:
#include<stdio.h>
#include<conio.h>

#include<malloc.h>
struct node
{
 int data;
 struct node *next;
};
struct node *top=NULL;

void push();
void pop();
void display();

void main()
{
int choice;
clrscr();
printf("\n PRPGRAM FOR STACK USLING LINKED LIST\N");\
while(1)
{
printf("\nENTER YOUR CHOICE\n 1.Push\n 2.Pop\n 3.Display\n 4.Exit\n");
scanf("%d",&choice);
switch (choice)
{
case 1: push(); break;
case 2: pop();  break;
case 3: display(); break;
case 4: exit(0);
 default: printf("\nWrong Choice\n");
}
}
}
void push()
{
struct node*new1;
int num;
new1=(struct node*)malloc(sizeof(struct node));
if(new1==NULL)
printf("\n Memory not created\n");
else
{
printf("\n Enter the data to be inserted\n");
scanf("%d",&num);
new1*data=num;
new1*next=top;
top=new1;
}
}
void pop()
{
int x;
struct node*temp;
if(top==NULL)
printf("\n Stack is empty\n");
else
{
x=top*data;
temp=top;
top=top*next;
free(temp);
printf("\n Deleted data is %d",x);
}
}
void display()
{
struct node*temp;
if(top==NULL)
printf("\n Stack is  empty\n");
else
{
temp=top;
while(temp!=NULL)
{
printf("%d",temp*data);
temp=temp*next;
}
}
}
```

```
if( top == MAX-1 )
return 1;
else
return 0;
}/*End of isFull*/
void display()
{
int i;
if( isEmpty() )
{
printf("\nStack is empty\n");
}
else
{
printf("\nStack elements :\n\n");
for(i=top;i>=0;i--)
printf(" %d\n", stack_arr[i] );
}
printf("\n");
}/*End of display()*/
```

//Experiment-3_part_B
Queue using array

```
//Roll No:        Class:          Div:            Name:
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 3
int queue_arr[MAX],rear=-1,front=-1;
void insert();
void del();
void display();
int isFull();
int isEmpty();
void main()
{
int choice,item;
clrscr();
while(1)
{
printf("\n1.Insert\n 2.Delete\n 3.Display\n 4.Exit\n");
scanf("%d",&choice);
switch(choice)
{
case 1:insert();    break;
case 2:del();       break;
case 3:display();   break;
case 4:exit(1);     break;
default:printf("\nWrong choice\n");
}
}
}
void insert()
{
int item;
if(front==-1)
front=0;
if(isFull())
{
printf("Queue overflow\n");
}
else
{
printf("Enter the element:");
scanf("%d",&item);
rear=rear+1;
queue_arr[rear]=item;
}
}
void del()
{
int item;
if(isEmpty())
{
printf("Queue is empty");
}
else
{
item=queue_arr[front];
front=front+1;
printf("Deleted element is %d",item);
}
```

//Experiment-7_part_B
Queue using

```
//Roll No:        Class:          Div:            Name:
#include<stdio.h>
#include<conio.h>
typedef struct queue
{
int info;
struct queue*next;
}queue;
struct queue*front=NULL,*rear=NULL;
void display();
void insert();
void delq();
void main()
{
int choice,pos,info;
char ch;
clrscr();
do
{
printf("enter your choice: \n1.insert \n 2.delete \n 3.display");
scanf("%d",&choice);
switch(choice)
{
case 1:insert();
       break;
case 2:delq();
       break;
case 3:display();
       break;
}
printf("\n do u wish to continue(y/Y)");
flushall();
scanf("%c",&ch);
}while(ch=='y'||ch=='Y');
}
void insert()
{
struct queue *temp;
temp=(queue*)malloc(sizeof(queue));
printf("enter info.  to be inserted");
scanf("%d",&temp->info);
temp->next=NULL;
if(front==NULL)
{
front=temp;
rear=temp;
}
else
{
rear->next=temp;
rear=rear->next;
}
}

void delq()
{
int element;
struct queue *temp;
if(front==NULL)
```

```
}
int isEmpty()
{
if(front==-1||front==rear+1)
return 1;
else
return 0;
}
int isFull()
{
if(rear==MAX-1)
return 1;
else
return 0;
}
void display()
{
int i;
if(isEmpty())
{
printf("Queue is empty");
return;
}
printf("Queue is:\n");
for(i=front;i<=rear;i++)
printf("%d\n",queue_arr[i]);
}
```

```
printf("queue empty can't delete");
else
{
element=front->info;
temp=front;
temp=NULL;
free(temp);
front=front->next;
printf("deleted element is %d",element);
}
}
void display()
{
struct queue *temp;
if(front==NULL)
printf("\n queue empty cant display");
else
{
for(temp=front;temp!=NULL;temp=temp->next)
printf("\n data is %d",temp->info);
}
}
```

Single link list -4

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
typedef struct node
{
 int data;
 struct node *next;
}node;
node * create();
node * insert_b(node *head); // void insert_b();
void insert_m(node *head);
node* insert_e(node *head);
node * delete_b(node *head);
void delete_m(node *head);
node* delete_e(node *head);
void *display(node *head);
// node *head=NULL;


int main()
{
 int i;
node *head=NULL;
 clrscr();
 do
 {
Clrscr();
  printf("\n1.create\n2.insert\n3.delete\n
4.display\
n5.display \n6.exit");
printf("\nenter your choice");
  scanf("%d",&i);
  switch(i)
  {
    case 1: head=create();   /// node * create();

        break;

    case 2: printf("\nenter your choice");
          printf("\n1.insert at the
beginning\n2.insert at the middle\
n3.insert at the end\n 4.exit");
          scanf("%d",&i);
          switch(i)
          {
             case 1: head=insert_b(head);
                  break;
          case 2: insert_m(head);
                  break;
          case 3: insert_e(head);
                  break;
                        case 4: exit(0);
          }
    case 3: printf("\nenter your choice");
          printf("\n1.delete at the
beginning\n2.delete at the middle
\n3.delete at the end");
          scanf("%d",&i);
          switch(i)
          {
            case 1: head=delete_b(head);
                  break;
          case 2: delete_m(head);
                  break;
          case 3: head=delete_e(head);
                  break;
                       case 4: exit(0);
          }
    case 4: display(head);
        break;

case 5: exit(0);
```

```
node * create()
{
 node *t1,*t,*head, *new;
int no;
 head=NULL;
 int i,no;
 printf("enter yor no of data or number
of nodes");
 scanf("%d",&no);
 for(i=1;i<=no;i++)
 {
new=(node*)malloc(sizeof(node));
  if(new==NULL)
{ printf("insufficient memory");
Return;
}
else
  {  printf("\nenter your data");
     scanf("%d",&(new->data));
    new->next=NULL;

     t=head;

   If(head==NULL)
      head=new;

    else
     {     while(t->next!=NULL)
            tmp=tmp->next;
          tmp->next=new;


      }

   } //ELSE end
 return(head);
}

node *insert_b(node *head)
{
node *t,*t1,*new;

new=(node*)malloc(sizeof(node));
printf("enter yor data");
scanf("%d",&(new->data));
t=head;
new->next=head;

head=new;
return(head);
}


Node * insert_e(node *head)
{
node *t,*t1,*new;
       int i,j;
       t=head;
       new=(node*)malloc(sizeof(node));
       printf("enter yor data");
       new->next=NULL;
       scanf("%d",&(new->data));

       if(head==NULL)
       { printf("empty ll");
         head=new;
        }
       Else
             {
       while(t1->next!=NULL)
             T1=t1->next;
       t1->next=new;
       new->next=NULL;


}
```

```
node *delete_b(node
*head)
 {
    node *tmp,*tmp1;

if(head==NULL)
    printf("empty cll");

else{
 tmp=head;

tmp1=head->next;
head=head->next;
//head=tmp1->next
tmp->next = NULL;
     free(tmp);
}
     return(head);
 }


  Void delete_e(node
  *head)
  {
 node *t,*t1;
 if(head==NULL) //case -I
 {
       printf("empty cll");
       return();
 }
 t=head;
 //casse- II- only one node

 if(head->next==NULL)
 {
       head=NULL;
       free(t);

 }
 //case III
else { T1=head;
 while(t1->next!=NULL)
 {     t=t1;
       t1=t1->next;
 }
  t-next=NULL;
  Free(t1);

  }
  }


  void *display(node
  *head)
  {
node *t=head;
printf("\n\n");
while(t!=NULL)
   printf("%d->",t->data);
  }


  Void delete_m(node
  *head)
  {
```

```
 }
 }while(i!=6);
getch();
}
```

```
Return(head);
}

Void insert_m(node *head)
{
node *t,*t1,*new;
        int i,j;
        t=head;
        new=(node*)malloc(sizeof(node));
        printf("ip position")
        scanf("%d",&pos);
        tpos=pos-1;
        while(tpos)
        {
                t1=t1->next;
                t=t1;
                tpos=tpos-1;
        }
        printf("enter yor data");
        scanf("%d",&(new->data));
        new->next=t1->next;
        t->next=new;

}
```

```
   node *t,*t1,*new;
int i,j;
t=head;
scanf("%d",&pos);
tpos=pos-1;
while(tpos)
{
        t1=t1->next;
        t=t1;
        tpos=tpos-1;
}

t->next=t1->next;
t1->next=NULL;
free(t1);

   }
```

//Experiment-5
Tree

```
//Roll No:        Class:          Div:            Name:
#include<stdio.h>
#include<conio.h>
//#include<malloc.h>

struct node{
        int data;
        struct node *left,*right;
        };

struct node *r=NULL,*new1;

void insert(struct node *r,struct node *nn);
void preorder(struct node *r);
void inorder(struct node *r);
void postorder(struct node *r);
void search();


void main()
{
        int ch;
        char m;
        clrscr();
        printf("\n program for implementation of binary search\n");
        printf("\n tree(bst) using link list \n");

do{
                printf("\n 1.creat bst\n 2.preorder traversal \n3.inorder
traversal\n4.postorder traversal \n5.search\n6.exit\n");

                printf("\n enter your choice \n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: do{
                                        new1=(struct node*)malloc(sizeof(struct node));
                                        printf("\n enter the data=>\n");
                                        scanf("%d",&new1->data);
                                        new1->left=null;
                                        new1->right=null;
                                        if(r==null)
                                          r=new1;
                                        else
                                          insert(r,new1);
                                        printf("\n do you want to continue (y/n)=>\n");
                                        m=getche();
```

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


void postorder(struct node *r)
{
 struct node *temp;
 temp=r;
 if(temp!=null)
        {
         postorder(temp->left);
         postorder(temp->right);
         printf("%d/t",temp->data);
         }
 }



void search()
{
  int key,flag=0;
  struct node *temp;
  temp=r;
  if(temp==null)
  printf("\n bst is empty search not possible \n");
  else{
        printf("\n enter the data to be searched \n");
        scanf("%d",&key);
        while(temp!=null)
        {
         if(key<temp->data)
          temp=temp->left;
         else
         temp=temp->right;
        if(temp->data==key)
          {
            flag=1;
            break;
          }
```

```
                                }while(m=='y'||m=='Y');
                                break;
                        case 2: preorder(r);
                                break;
                        case 3: inorder(r);
                                break;
                        case 4: postorder(r);
                                break;
                        case 5: search();
                                break;
                        }
                }while(ch!=6);
        }


void insert(struct node *r,struct node *nn)
{       if(nn->data<r->data)
        {
         if(r->left==NULL)
            r->left=nn;
         else
             insert(r->left,nn);
         }
        else
        { if(r->right==NULL)
             r->right=nn;
          else
             insert(r->right,nn);
        }
}


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

```
        }
  if(flag==1)
        printf("\n node %d is present in bst .\n",key);
  else  printf("\n node %d is absent in bst \n",key);
  }
}
```

EXP no -6

Implement Graph using adjacency Matrix with BFS & DFS traversal.

```
#include<stdlib.h>
int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];
int delete();
void add(int item);
void bfs(int s,int n);
void dfs(int s,int n);
void push(int item);
int pop();

void main()
{
int n,i,s,ch,j;
char c,dummy;
printf("ENTER THE NUMBER VERTICES ");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("ENTER 1 IF %d HAS A NODE WITH %d ELSE 0 ",i,j);
scanf("%d",&a[i][j]);
}
}
printf("THE ADJACENCY MATRIX IS\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf(" %d",a[i][j]);
}
printf("\n");
}

do
{
for(i=1;i<=n;i++)
vis[i]=0;
printf("\nMENU");
printf("\n1.B.F.S");
printf("\n2.D.F.S");
printf("\nENTER YOUR CHOICE");
scanf("%d",&ch);
printf("ENTER THE SOURCE VERTEX :");
scanf("%d",&s);

switch(ch)
{
case 1:bfs(s,n);
break;
case 2:
dfs(s,n);
break;
}
printf("DO U WANT TO CONTINUE(Y/N) ? ");
scanf("%c",&dummy);
scanf("%c",&c);
}while((c=='y')||(c=='Y'));
}//main exit

void dfs(int s,int n)
{
int i,k;
push(s);
vis[s]=1;
k=pop();
if(k!=0)
printf(" %d ",k);
while(k!=0)
{
for(i=1;i<=n;i++)
if((a[k][i]!=0)&&(vis[i]==0))
```

```
void bfs(int s,int n)
{
int p,i;
add(s);
vis[s]=1;
p=delete();
if(p!=0)
printf(" %d",p);
while(p!=0)
{
for(i=1;i<=n;i++)
if((a[p][i]!=0)&&(vis[i]==0))
{
add(i);
vis[i]=1;
}
p=delete();
if(p!=0)
printf(" %d ",p);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
bfs(i,n);
}

void add(int item)
{
if(rear==19)
printf("QUEUE FULL");
else
{
if(rear==-1)
{
q[++rear]=item;
front++;
}
else
q[++rear]=item;
}
}
int delete()
{
int k;
if((front>rear)||(front==-1))
return(0);
else
{
k=q[front++];
return(k);
}
}
```

```
{
push(i);
vis[i]=1;
}
k=pop();
if(k!=0)
printf(" %d ",k);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
dfs(i,n);
}

void push(int item)
{
if(top==19)
printf("Stack overflow ");
else
stack[++top]=item;
}

int pop()
{
int k;
if(top==-1)
return(0);
else
{
k=stack[top−];
return(k);
}
}
```

GROUP B

Polynomial addition – EXP NO -8

```
#include <stdio.h>
typedef struct pnode
{
float coef;
int exp;
struct pnode *next;
}p;

p *getnode();

void main()
{
p *p1,*p2,*p3;

//p *getpoly(),*add(p*,p*);

//void display(p*);
clrscr();
printf("\n enter first polynomial");
p1=getpoly();
printf("\n enter second polynomial");
p2=getpoly();
printf("\nthe first polynomial is");
display(p1);
printf("\nthe second polynomial is");
display(p2);
p3=add(p1,p2);
printf("\naddition of two polynomial is :\n");
display(p3);

}
p *getpoly()//struct pnode *getpoly()
{
p *temp,*New,*last;
int flag,exp;
char ans;
float coef;
temp=NULL;
flag=1;//head
printf("\nenter the polynomial in descending order of exponent");
do
{
printf("\nenter the coef & exponent of a term");
scanf("%f%d",&coef,&exp);
New=getnode();
if(New==NULL)
printf("\nmemory cannot be allocated");
New->coef=coef;
New->exp=exp;
if(flag==1)
{
temp=New;
last=temp;
flag=0;
}
else
{
last->next=New;
last=New;
}
printf("\ndou want to more terms");
ans=getch();
}
while(ans=='y');
return(temp);
}


p *getnode()
{
p *temp;
temp=(p*) malloc (sizeof(p));
temp->next=NULL;
return(temp);}
```

```
void display(p*head)
{
p*temp;
temp=head;
if(temp==NULL)
printf("\npolynomial empty");
while(temp->next!=NULL)
{
printf("%0.1fx^%d+"  ,temp->coef,temp->exp);
temp=temp->next;
}
printf("\n%0.1fx^%d",temp->coef,temp->exp);
getch();
}

p*add(p*first,p*second)
{
p *p1,*p2,*temp,*dummy;
char ch;
float coef;
p *append(int,float,p*);
p1=first;
p2=second;
temp=(p*)malloc(sizeof(p));
if(temp==NULL)
printf("\nmemory cannot be allocated");
dummy=temp;
while(p1!=NULL&&p2!=NULL)
{
if(p1->exp==p2->exp)
{
coef=p1->coef+p2->coef;
p1=p1->next;
p2=p2->next;
p3->coef=coef;
p3->exp=p2->exp;
p3=p3->next;
}
Else if(p1->exp < p2->exp)
{
 p2=p2->next;
p3->coef= p2->coef;
p3->exp=p2->exp;
p3=p3->next;

}
else
if(p1->exp>p2->exp)
{
p3->coef= p1->coef;
p3->exp=p1->exp;
p3=p3->next;
p1=p1->next;

}
while(p1!=NULL)
{
p3->coef= p1->coef;
p3->exp=p1->exp;
p3=p3->next;
p1=p1->next;
}
while(p2!=NULL)
{
p2=p2->next;
p3->coef= p2->coef;
p3->exp=p2->exp;
p3=p3->next;
}
temp->next=NULL;
temp=dummy->next;
free(dummy);
return(temp);
}
```

EXP no- 9
Postfix evaluation

```
#include<stdio.h>
int stack[20];
int top = -1;

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

int pop()
{
    return stack[top--];
}

int main()
{
    char exp[20];
   char *e;
    int n1,n2,n3,num;
    printf("Enter the expression :: ");
    scanf("%s",exp);
    e = exp;
    while(*e != '\0')
    {
        if(isdigit(*e))
        {
            num = *e - 48;
            push(num);
        }
        else
        {
            n1 = pop();
            n2 = pop();
            switch(*e)
            {
            case '+':
            {
                n3 = n1 + n2;
                break;
            }
            case '-':
            {
                n3 = n2 - n1;
                break;
            }
            case '*':
            {
                n3 = n1 * n2;
                break;
            }
            case '/':
            {
                n3 = n2 / n1;
                break;
            }
            }
            push(n3);
        }
        e++;
    }
    printf("\nThe result of expression %s  =  %d\n\n",exp,pop());
    return 0;
}
```

Group C

Single circular link list - 10

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
typedef struct node
{
 int data;
 struct node *next;
}node;
node * create();
node *insert_b(node *head); // void insert_b();
insert_m(node *head);
insert_e(node *head);
node *delete_b(node *head);
delete_m(node *head);
delete_e(node *head);
void *display(node *head);
// node *head=NULL;


int main()
{
 int i;
node *head=NULL;
 clrscr();
 do
 {
Clrscr();
  printf("\n1.create\n2.insert\n3.delete\n4.display\
n5.display \n6.exit");
printf("\nenter your choice");
  scanf("%d",&i);
  switch(i)
  {
    case 1: head=create();   /// node *create();

        break;

    case 2: printf("\nenter your choice");
          printf("\n1.insert at the beginning\n2.insert at the middle\n3.insert at the end\n 4.exit");
          scanf("%d",&i);
          switch(i)
          {
            case 1: head=insert_b(head);
                  break;
          case 2: insert_m(head);
                  break;
          case 3: insert_e(head);
                  break;
                        case 4: exit(0);
          }
    case 3: printf("\nenter your choice");
          printf("\n1.delete at the beginning\n2.delete at the middle\n3.delete at the end");
          scanf("%d",&i);
          switch(i)
          {
            case 1: head=delete_b(head);
                  break;
          case 2: delete_m(head);
                  break;
          case 3: delete_e(head);
```

```
node * create()
{
 node *t1,*t,*head, *new;
int no;
 head=NULL;
 int i,no;
 printf("enter yor no of data or number of nodes");
 scanf("%d",&no);
 for(i=1;i<=no;i++)
 {
new=(node*)malloc(sizeof(node));
  if(new==NULL)
{ printf("insufficient memory");
Return;
}
else
  {  printf("\nenter your data");
     scanf("%d",&(new->data));
    new->next=NULL;

     t=head;

  If(head==NULL)
     { head=new;
        New->next=new;
}

    else
     {     while(t->next!=head)
            tmp=tmp->next;
          tmp->next=new;
          new->next = head;

      }

  } //ELSE end
 return(head);
}

node *insert_b(node *head)
{
node *t,*t1,*new;

new=(node*)malloc(sizeof(node));
printf("enter yor data");
scanf("%d",&(new->data));
t=head;
t1=head->next.

While (t1->next!=head)
T1=t1->next;

T1->next=new

head=new;
return(head);
}


Node * insert_e(node *head)
{
node *t,*t1,*new;
       int i,j;
       t=head;
       new=(node*)malloc(sizeof(node));
       if(head==NULL)
       { printf("empty ll");
          head=new;
        t=t->next;
       return(head);}
       Else
              { t1=head->next;
```

```
   node * create()
   {
    node *t1,*t,*head, *new;
   int no;
    head=NULL;
    int i,no;
printf("enter yor no of data or number of nodes");
    scanf("%d",&no);
    for(i=1;i<=no;i++)
    {
     if(head==NULL) {
   new=(node*)malloc(sizeof(node));

      head=new; head->next=head;
   }
    else
    {
 new=(node*)malloc(sizeof(node));
   if(new==NULL)
  {   printf("error in creating node");
       Exit(0);
   }
    tmp=head;
  if(tmp->next==tmp)
       { tmp->next= new;
          New->next=head; }
 else
  {     while(tmp->next!=head)
         tmp=tmp->next;
       tmp->next=new;
       new->next=head;
   }

  printf("\nenter your data");
  scanf("%d",&(new->data));
 } //for loop end
 return(head);
}

node *insert_b(node *head)
{
node *t,*t1,*new;

new=(node*)malloc(sizeof(node));
printf("enter yor data");
scanf("%d",&(new->data));
t=head;
t1=head->next.

While (t1->next!=head)
T1=t1->next;

T1->next=new

head=new;
return(head);
}


Node * insert_e(node *head)
{
node *t,*t1,*new;
       int i,j;
       t=head;
       new=(node*)malloc(sizeof(node));
       if(head==NULL)
       { printf("empty ll");
          head=new;
        t=t->next;
       return(head);}
       Else
              { t1=head->next;
```

```
                    break;
                        case 4: exit(0);
            }
    case 4: display(head);
        break;

case 5: exit(0);
  }
 }while(i!=6);
getch();
}
```

```
        while(t1->next!=head)
        {
              T1=t1->next;
        }
        printf("enter yor data");
        scanf("%d",&(new->data));
        t1->next=new;
        new->next=head;


}

Void insert_m(node *head)
{
node *t,*t1,*new;
        int i,j;
        t=head;
        new=(node*)malloc(sizeof(node));
        printf("ip position")
        scanf("%d",&pos);
        tpos=pos-1;
        while(tpos)
        {
              t1=t1->next;
              t=t1;
              tpos=tpos-1;
        }
        printf("enter yor data");
        scanf("%d",&(new->data));
        new->next=t1->next;
        t->next=new;

}
```

```
        while(t1->next!=head)
        {
              T1=t1->next;
        }
        printf("enter yor data");
        scanf("%d",&(new->data));
        t1->next=new;
        new->next=head;


}

Void insert_m(node *head)
{
node *t,*t1,*new;
        int i,j;
        t=head;
        new=(node*)malloc(sizeof(node));
        printf("ip position")
        scanf("%d",&pos);
        tpos=pos-1;
        while(tpos)
        {
              t1=t1->next;
              t=t1;
              tpos=tpos-1;
        }
        printf("enter yor data");
        scanf("%d",&(new->data));
        new->next=t1->next;
        t->next=new;

}
```