Program. 1

```
#include<iostream>
#include<cstring>
#include<math.h>
using namespace std;
int main()
{
int option;
int a[10],b[10],c[10],m,n,i,j;
void union1(int a[10],int b[10],int m,int n);
void intersection(int a[10],int b[10],int m,int n);
void difference(int a[10],int b[10],int m,int n);
void symdiff(int a[10],int b[10],int m,int n);
do
{
cout<<"\n1. READ";
cout<<"\n2. UNION";
cout<<"\n3. INTERSECTION";
cout<<"\n4. DIFFERENCE";
cout<<"\n5. SYM. DIFFERENCE";
cout<<"\n6. EXIT";
cout<<"\n\t\t Enter your option";
cin>>option;
switch(option)
{
case 1: cout<<"\n.....Enter No of elements in set A\t";
cin>>m;
cout<<"\n.....Enter %d elements for set A\t",m;
for(i=0;i<m;++i)
cin>>a[i];
cout<<"\n.....Enter no of ements in set B\t";
cin>>n;
cout<<"\n.....Enter %d elements of set B\t",n;
for(i=0;i<n;++i)
cin>>b[i];
 break;
 case 2: union1(a,b,m,n);
 break;
 case 3: intersection(a,b,m,n);
 break;
case 4:difference(a,b,m,n);
 break;
case 5: symdiff(a,b,m,n);
 case 6: break;
```

```
}
}while(option<6);
}
void union1(int a[10],int b[10],int m,int n)
{ int i,j,k,c[10],x;
for(i=0;i<m;++i)
c[i]=a[i];
k=m;
//Select uncommon element of set B
for(j=0;j<n;++j)
{ x=0;
for(i=0;i<m;++i)
{ if(b[j]==a[i])
{
x=1;
break;
}
}
if(x==0)
{ c[k]=b[j];
k++;
}
 }
cout<<("\n.....UNION OF SET A & set B\t");
for(i=0;i<k;++i)
cout<<"\t c[i]";
}
void intersection(int a[10],int b[10],int m,int n)
{ int i,j,k,c[10],x;
 // Select uncommon from set A or set B
k=0;
for(j=0;j<n;++j)
{ x=0;
for(i=0;i<m;++i)
{
if(b[j]==a[i])
{
x=1;
break;
}
}
if(x==1)
{ c[k]=b[j];
k++;
}
}
```

```
cout<<"\n.....INTERSECTION A & set B\t";
for(i=0;i<k;++i)
cout<<"\t"<<c[i];
}
void difference(int a[10],int b[10],int m,int n)
{
 int i,j,k,c[10],x;
 k=0;
 for(i=0;i<n;++i)
{ x=0;
for(j=0;j<m;++j)
{ if(a[i]==b[j])
{ x=1;
break;
}
}
if(x==0)
{ c[k]=a[i];
k++;
}
}
cout<<"\n.....DIFFERENCE A-B is\t";
for(i=0;i<k;++i)
cout<<"\t"<<c[i];
}
void symdiff(int a[10],int b[10],int m,int n)
{ int i,j,k=0,c[10],x,z=0;
 k=0;
for(i=0;i<m;i++)
{ x=0;
for(j=0;j<n;j++)
{ if(a[i]==b[j])
{ x=1;
break;
}
}
 if(x==0)
{ c[k]=a[i];
 k++;
} }
z=k;
for(j=0;j<m;++j)
{ x=0;
for(i=0;i<n;++i)
{
if(b[j]==a[i])
```

```
{
x=1;
break;
}
}
if(x==0)
{ c[z]=b[j];
z++;
}
}cout<<"\n.....S.DIFFERENCE is\t";
for(i=0;i<z;++i)
cout<<"\t"<<c[i];
}

//OUTPUT
1.READ
2.UNION
3. INTERSECTION
4. DIFFERENCE
5. SYM. DIFFERENCE
6. EXIT
        Enter your option1
.....Enter No of elements in set A  2
.....Enter 2 elements for set A     1 2
.....Enter no of elements in set B    2
.....Enter 2 elements of set B      1 3
1.READ
2.UNION
3. INTERSECTION
4. DIFFERENCE
5. SYM. DIFFERENCE
6. EXIT
        Enter your option2
.....UNION OF SET A & set B 1 2          3
1.READ
2.UNION
3. INTERSECTION
4. DIFFERENCE
5. SYM. DIFFERENCE
6. EXIT
        Enter your option3



.....INTERSECTION  OF SET A & set B  1
```

.....

1.READ
2.UNION
3. INTERSECTION
4. DIFFERENCE
5. SYM. DIFFERENCE
6. EXIT

Enter your option4
.....DIFFERENCE OF SET A & set B 2

1.READ
2.UNION
3. INTERSECTION
4. DIFFERENCE
5. SYM. DIFFERENCE
6. EXIT

Enter your option5
.....DIFFERENCE OF SET A & set B 2 3

.....

Program: 2

```
#include<iostream>
#define 10 10 void
main()
{
        int option;
       int a[10][10],b[10][10],z[10][10],i,j,k,r1,c1,r2,c2; void
addm(int a[10][10] , int b[10][10], int r1,int c1,int r2,int c2); void
multm(int a[10][10] , int b[10][10], int r1,int c1,int r2,int c2);
        void transm(int a[10][10] , int r1,int c1);
       void saddlepoint(int a[10][10],int r1,int c1);
        do
        {
                cout<<"\n1. READ";
               cout<<"\n2. ADDITION";
                cout<<"\n3. MULTIPLICATION";
                cout<<"\n4. TRANSPOSE";
        cout<<“\n 5. SADDLEPOINT”;
                cout<<"\n 6. EXIT";
                cout<<"\n\t\t Enter your option";
                cin>>ption;
switch(option)
                {
                case 1: cout<<"\n.....Enter size of  row & col of A matrix";
                        cin>>r1>>c1;
                        cout<<"\n.....Enter size of  row & col of B matrix";
                        cin>>r2>>c2;
                        cout<<"\n.....Enter the elements of matrix A rowwise\t";
                        for(i=0;i<r1;++i)
                        {
                                for(j=0;j<c1;++j)
                                {
                                        cin>>a[i][j];
                                 }
                        }
                        cout<<"\n.....Enter the elements of matrix B rowwise\t";
                        for(i=0;i<r2;++i)
                        {
                                for(j=0;j<c2;++j)
                                {
                                        cin>>b[i][j];
                                 }
                        }
                       break;
        case 2: addm(a,b,r1,c1,r2,c2);
        break;          case 3:
multm(a,b,r1,c1,r2,c2);
```

```
                                        break;
                        case 4: transm(a,r1,c1);
                                break;
                        case 5:saddlepoint(a,r1,c1);
                         break;
case 6: break;

                        }
                }while(option<4);
                getch();
}

void addm(int a[10][10] , int b[10][10], int r1,int c1,int r2,int c2)
{                       int i,j,k,z[10][10];
                // addition with pointer
                         if(r1!= r2 || c1!=c2 )
                                {
                                        cout<<(" Cannot be added");

                                }

                          else
                        {

                                  for(i=0;i<r1;++i)
                        {         for(j=0;j<c1;++j)
                                  {
                                    z[i][j]=a[i][j]+b[i][j];
                                  }
                        }
                        cout<<("\nThe addition of given matrices is");
                        for(i=0;i<r1;++i)
                        {
                                  for(j=0;j<c1;++j)
                                  {
                                    cout<<"\t"<<z[i][j];
                                  }
                            cout<<"\n";
                        }
                        }
}


void multm(int a[10][10] , int b[10][10], int r1,int c1,int r2,int c2)
{                       int z[10][10],i,j,k;
                 if(c1!=r2)
                        {
```

```
                cout<<"\n\tRows of first and Columns of second are not
matching";

                }
                else
        {
                for(i=0;i<r1;++i)
                {   for(j=0;j<c2;++j)
                        {       z[i][j]=0;
                                for(k=0;k<r2;++k)
                                        z[i][j]=a[i][k]*b[k][j]+z[i][j];
                        }
                }
                cout<<"\n The matrix multiplication is";
          for(i=0;i<r1;++i)                     {
cout<<("\n");
                        for(j=0;j<c2;++j)
                        {
                          cout<<"\t"<<z[i][j];
                        }
                }
                }

}
 void transm(int a[10][10],int r1,int c1)
{   int i,j,k;

if(r1!= c1)
                        {
                                 cout<<" Row and column must be same ";

                        }
else
{
                cout<<"\n Transpose is:";
                for(i=0;i<c1;++i)
                {
                        for(j=0;j<r1;++j)
                        {
                                cout<<"\t"<< a[j][i];
                        }
                                cout<<"\n";
                }
      }
}

void saddlepoint(int a[10][10],int r1,int c1)
```

```
{

      int i, j, small, large, col_of_small, row_of_large;

for(i=0;i<m;i++)                          // find saddle point row wise
{

     small=a[i][0];
col_of_small=0;
for(j=1;j<r1;j++)
               {
                   if(a[i][j]<small)
                     {
small=a[i][j];
col_of_small=j;
                     }

                                                                  //  find  the  largest element
in col_of_small
                     large=a[0][col_of_small];
for(j=1;j<c1;j++)
                   if(a[j][col_of_small]>large)
                          {
                              large=a[0][col_of_small];
row_of_large=j;
                          }

                   if(i==row_of_large)
                     {
  cout<<"\n Saddle point exist at (%d%d) with values as, i, col_of_small,a[i][col_of_samll];
return(1);
} }
cout<<"\n Saddle point does not exist"; return(0);
}


//OUTPUT

1. READ
2. ADDITION
3. MULTIPLICATION
4. TRANSPOSE
5. SADDLE POINT
6. EXIT
          Enter your option    1
.....Enter No of rows and no of columns A   2 2
```

.....Enter No of rows and no of columns B 2 2
.....Enter the elements of matrix A row wise 1 1 1 1
.....Enter the elements of matrix A row wise 1 1 1 1

1. READ
2. ADDITION
3. MULTIPLICATION
4. TRANSPOSE
5. SADDLE POINT
6. EXIT

Enter your option 2

The addition of given matrices is
2 2
2 2

1. READ
2. ADDITION
3. MULTIPLICATION
4. TRANSPOSE
5. SADDLE POINT
6. EXIT

Enter your option 3

The matrix multiplication is
2 2
2 2

1. READ
2. ADDITION
3. MULTIPLICATION
4. TRANSPOSE
5. SADDLE POINT
6. EXIT

Enter your option 4

.....Enter No of rows and no of columns A 2 2
.....Enter the elements of matrix A row wise 1234
Transpose is
1 3
2 4

1. READ
2. ADDITION
3. MULTIPLICATION
4. TRANSPOSE
5. SADDLE POINT

6. EXIT

Enter your option 5

.....Enter No of rows and no of columns A 2 2

.....Enter the elements of matrix A row wise 1234 Saddle point exist at with (1,0) values as 3

Assignment No.3

Title: Write a program to implement a) Sort the set of strings in ascending order using Bubble sort and descending order by using Selection sort. b) Search for particular string using binary search.

Program:

```
#include <iostream>
#include <string> typedef
struct mobile
{
 long  mobileno;  char
name[20];
        float  billamt;
}mobile;
int  binsearch(mobile st[],char uname[],int n);
void print(mobile st[],int n); void
read(mobile  st[],int n); void
selectionsort(mobile a[],int n);
void bubblesort(mobile  a[],int n);

void main()
{
        mobile st[30];
char uname[20];
        int n,i,op,position;
        do
        {
                flushall();
                cout<<"\n1)CREAT\n2)PRINT";
                cout<<"\n3)Display the data in ascending order of name(Bubble sort)";
cout<<"\n4)Display the data in descending order of name(Selection sort)";
cout<<"\n5)Display details for user name specified by user(binary search)";
cout<<"\n6)Quit";

                cout<<"\nEnter Your Choice:";
                cin>>op;

                switch(op)
                {
                        case 1: cout<<"\nEnter No. of Users :";
                                cin>>n;
                                read(st,n);
                break;

                        case 2: print(st,n);
                                break;
```

```
                        case 3: bubblesort(st,n);
                                print(st,n);
                        case 4: selectionsort(st,n);
        print(st,n);break;
                        case 5: cout<<"\nPlease ensure that data is sorted using option no.3";
                                cout<<"\nEnter user name to search details: ";
                                cin>>uname;

                                position=binsearch(st,uname,n);

                                if(position==-1)
                cout<<"\nnot found";
                                else
                                {
                                        cout<<"\n found at location=%d",position+1;

         cout<<"\n%s\t%ld\t%6.2f",st[position].name,st[position].mobileno,st[position].billamt;
                                }
                                break;
                }
        }while(op!=6);
}
int binsearch(mobile  st[],char uname[],int n)
{
        int i,j,k,comp=0;
        i=0;
j=n-1;
while(i<=j)
        {
                k=(i+j)/2;
comp++;

                if(strcmp(uname,st[k].name) == 0)
                {
                        cout<<"\nNo. of comparisons = %d ",comp;
        return(k);
                }
                else
                        if(strcmp(uname,st[k].name) < 0
)                               j=k-1;                  else
                        i=k+1;
        }
                cout<<"\nNo. of comparisons = %d ",comp;
return(-1);
}
void print(mobile  st[],int n)
{
```

```
        int i;
for(i=0;i<n;i++)
cout<<"\n%20s %15ld
%6.2f",st[i].name,st[i].mobil
eno,st[i].billamt;
}
void read(mobile st[],int n)
{
        int i;
        float billamt;
        long mobileno;

        cout<<"\n enter data(name mobile No.(9 disgits 10) Bill Amount ): \n";

        for(i=0;i<n;i++)
        {
                cout<<"Enter data for %d student: ",i+1;
cin>>st[i].name>>mobileno>>billamt;                st[i].billamt=billamt;
                st[i].mobileno=mobileno;
        }
}
void selectionsort(mobile  a[],int n)
{
        int i,j,k,passes=0,comp=0;
        mobile temp;

        for(i=0;i<n-1;i++)
        {
                passes++;
        k=i;
                for(j=i+1;j<n;j++)
                {
                        comp++;
                        if(strcmp(a[j].name,a[k].name) > 0)
                                k=j;
                }
                temp=a[i];
a[i]=a[k];
                a[k]=temp;
        }
        cout<<("\nPasses = %d\t Comparisons = %d",passes,comp);
}

void bubblesort(mobile  a[],int n)
{
        int i,j,k,passes=0,comp=0;
        mobile temp;
```

```
        for(i=1;i<n;i++)
        {       passes++;
                  for(j=0;j<n-1;j++)
                  {
                            comp++;
                            if(strcmp(a[j].name,a[j+1].name) > 0)
                            {
                                      temp=a[j];
        a[j]=a[j+1];
a[j+1]=temp;
                            }                   }
        }
        cout<<"\nPasses = %d\t Comparisons = %d",passes,comp;
}
//OUTPUT
1)create
2)print
3)Display the data in ascending order of name(Bubble sort)
4)Display the data in descending order of name(Selection sort)
5)Display details for user name specified by user(binary search)
6)Quit
Enter Your Choice:1

Enter No. of Users :2 enter data(name mobile No.(9
disgits 10) Bill Amount ):
Enter data for 1 student: AAA     123456   1000
Enter data for 2 student: BBB      123456    2000
1)create
2)print
3)Display the data in ascending order of name(Bubble sort)
4)Display the data in descending order of name(Selection sort)
5)Display details for user name specified by user(binary search)
6)Quit
Enter Your Choice:2
AAA         123456      1000.00
BBB         123456       2000.00
1)create
2)print
3)Display the data in ascending order of name(Bubble sort)
4)Display the data in descending order of name(Selection sort)
5)Display details for user name specified by user(binary search)
6)Quit
Enter Your Choice:3
Passes = 1       Comparisons = 1
AAA         123456       1000.00
```

BBB 123456 2000.00
1)create
2)print
3)Display the data in ascending order of name(Bubble sort)
4)Display the data in descending order of name(Selection sort)
5)Display details for user name specified by user(binary search)
6)Quit
Enter Your Choice:4
Passes = 1 Comparisons = 1
BBB 123456 2000.00
AAA 123456 1000.00
1)create
2)print
3)Display the data in ascending order of name(Bubble sort)
4)Display the data in descending order of name(Selection sort)
5)Display details for user name specified by user(binary search)
6)Quit
Enter Your Choice:5
Please ensure that data is sorted using option no.3
Enter user name to search details: BBB

No. of comparisons = 1
found at location=1

Assignment No.4

Title: Write a program to Implement stack as a ADT and perform following operation 1. Infix to postfix expression 2. Evaluation of postfix expression.

Program:

```
#include<iostream>
#include<cstring>
#include<math.h> using
namespace std;
class stack
{ private :            typedef
struct stack1
        {
int data;
        struct stack1 *next;
        }node;
node *top;
public:    stack();
        ~stack();
        void push(char,node **);
int pop(node **);          int
stackempty(node *); void
postfix(); void eval();
        };

int main()
{ int ch;
stack st;
do {
cout<<"\n1.Infix to postfix \n2.Evaluation of postfix exp. \n3. Exit";
cout<<"\n Enter your choice"; cin>>ch; switch(ch)
{
case 1: st.postfix();
break; case 2:
st.eval();
break; case
3:break;
}
}while(ch!=3);
}

stack::stack()
{
top=NULL;
}
```

```
stack :: ~stack() {
node *temp;
temp=top;
if(temp==NULL)
delete temp; else
{
while(temp!=NULL)
{ temp=temp-
>next;
top=NULL;
top=temp; }
delete temp; } }
void stack :: push(char x,node **top)
{ node *newnode;
newnode=new node;
newnode->next=NULL;
newnode->data=x;
newnode->next=*top;
*top=newnode;
} int stack::pop(node
**top)
{ int
x;
node *temp;
x=(*top)->data;
temp=*top;
*top=(*top)->next;
delete temp; return
x; }

int stack :: stackempty(node *temp)
{
if(temp==NULL)
return 1;
else
return 0; }
void stack::postfix() { char
exp[40]; int length,j,l; char
ch; char post[40]; j=0;
cout<<"\n enter exp";
cin>>exp;
length=strlen(exp);
cout<<"\n\n The postfix is:";
for(int i=0;i<=length-1;i++)
{ ch=exp[i];
if(ch=='('||ch=='+'||ch=='-'||ch=='*'||ch=='/')
```

```
push(ch,&top);
else if(ch!=')')
{ post[j]=ch;
j++; cout<<ch;
} else { do
{ ch=pop(&top);
if(ch!='(') {
post[j]=ch;
j++;
cout<<ch;
}
}while(ch!='(');
} }
post[j]='$';
}

void stack::eval() {
char exp[40]; int
length,val,a,b,c,l;
char ch; char
post[40]; int j=0,i=0;
cout<<"\n enter exp";
cin>>exp;
length=strlen(exp);
exp[length]='$';
for(i=0;i<=length;i+
+)
{ ch=exp[i];
if(ch=='^'||ch=='+'||ch=='-'||ch=='*'||ch=='/')
{ b=pop(&top);
a=pop(&top); switch(ch) {
case '+': c=a+b;break; case
'-': c=a-b;break; case '*':
c=a*b;break; case '/':
c=a/b;break; case '^':
c=pow(a,b);break;
} push(c,&top);
} else
if(ch!='$') {
val=ch-96;
push(val,&top);
}
} c=pop(&top);
cout<<"\n result:"<<c;
}
```

Output:

1. Infix to postfix 2.
Evaluation of postfix exp.
3. Exit
Enter your choice 1
Enter Infix Exp: (a+b)
Postfix Exp is:ab+

1. Infix to postfix 2.
Evaluation of postfix exp.
3. Exit Enter your choice 2
ab+ Value: 3

Assignment No.5

Title: Program to implement Circular Queue.

Program:

```
#include <iostream> using
namespace std; int
cqueue[5]; int front = -1,
rear = -1, n=5; void
insertCQ(int val) {
  if ((front == 0 && rear == n-1) || (front == rear+1)) {
    cout<<"Queue Overflow \n";
    return;   }    if
(front == -1) {
front = 0;      rear =
0;   } else {      if
(rear == n - 1)
rear = 0;      else
rear = rear + 1;
  }
  cqueue[rear] = val ;
} void deleteCQ()
{
  if (front == -1) {
cout<<"Queue Underflow\n";
    return ;
  }
  cout<<"Element deleted from queue is : "<<cqueue[front]<<endl;

  if (front == rear) {
front = -1;      rear =
-1;   } else {      if
(front == n - 1)
front = 0;      else
    front = front + 1;
  } } void
displayCQ() {
  int f = front, r = rear;
  if (front == -1) {
cout<<"Queue is empty"<<endl;
    return;
  }
  cout<<"Queue elements are :\n";
  if (f <= r) {      while (f
<= r){
cout<<cqueue[f]<<" ";
f++;      }   } else {
while (f <= n - 1) {
```

```
cout<<cqueue[f]<<" ";
f++;      }      f = 0;
while (f <= r) {
cout<<cqueue[f]<<" ";
f++;      }
  }
  cout<<endl;
} int main()
{
  int ch, val;    cout<<"1)Insert\n";
cout<<"2)Delete\n";
cout<<"3)Display\n";
cout<<"4)Exit\n";    do {
cout<<"Enter choice : "<<endl;
     cin>>ch;
switch(ch) {
case 1:
        cout<<"Input for insertion: "<<endl;
        cin>>val;
insertCQ(val);
break;          case 2:
deleteCQ();
break;          case 3:
displayCQ();
break;          case 4:
        cout<<"Exit\n";
break;
        default: cout<<"Incorrect!\n";
     }
  } while(ch != 4);
return 0; }
```

Output

```
1)Insert
2)Delete
3)Display
4)Exit Enter
choice : 1 Input
for insertion:
Enter choice : 1 Input
for insertion:
Enter choice : 1 Input
for insertion:
Enter choice : 1 Input
for insertion:
Enter choice : 1 Input
for insertion:
Enter choice : 2
```

Element deleted from queue is : 5
Enter choice : 2
Element deleted from queue is : 3
Enter choice : 2
Element deleted from queue is : 2
Enter choice : 1
Input for insertion: 6
Enter choice : 3 Queue
elements are :
7 9 6
Enter choice : 4
Exit

Assignment No 6

Title: Construct an Expression Tree from postfix and prefix expression. Perform recursive and non- recursive In-order, pre-order and post-order traversals.

Program:

```
#include<iostream>
#include<ctype.h>
#define size 20 using
namespace std;
class btree
{ public: char
data;
        btree *left;
btree *right;
};

class EXP_TREE
{
btree *stack[size]; int
top;
public: btree *root;
EXP_TREE();
void create(char exp[]); void
push(btree *);
void push1(btree *item, int *top, btree *s[10]);
void inorder(btree *root);  void preorder(btree
*root);
void postorder(btree *root);

void nonrecinorder(btree *root); void
nonrecpreorder(btree *root);
void nonrecpostorder(btree *root);

btree* pop();
void pop1(int *top,btree *s[10],btree **current); int
stempty1(int);
};

EXP_TREE::EXP_TREE()
{
root=NULL;

top=-1;
}
```

```
void EXP_TREE::create(char exp[])
{ int pos; char ch; pos=0; ch=exp[pos];
while(ch!=‟\0‟) { root=new btree; root-
>left=root->right=NULL; root->data=ch;
if(isalpha(ch)) push(root); elase
if(ch==‟+‟||ch==‟-„||ch==‟*‟||ch==‟/‟)
root->right=pop(); root->left=pop();
push(root); } else cout<<”Invalid character
in expression”;
pos++;
ch=exp[pos];
} root=pop();
}
void EXP_TREE :: push(btree *Node)
{ If(top+1>=size)
cout<<”Error: Stack is full”;
top++; stack[top]=Node; }
btree *EXP_TREE :: pop()
{ btree *Node; if(top==-1)
cout<<”Error:Stack is empty”;
Node=stack[yop]; top--; return(Node);
} void EXP_TREE :: inorder(btree
*root)
{ btree *temp;
temp=root;
if(temp!=NULL)
{inorder(temp->left);
cout<<””<<temp->data;
inorder(temp->right);
}
}
void EXP_TREE::nonrecinorder(btree *root)
{ btree *current,
*s[10]; int top1=-1;
if(root==NULL)
{ cout<<”\n Tree is
empty”;
return; }
Current=root;
for(;;)
{ while(current!=NULL)
{
push1(current,&top,s);
current=current->left }
If(!stempty1(top1))
{ pop1(&top1,s,&current);
cout<<””<<current->data;
current=current->right;
```

```
} else
return;
} }

void EXP_TREE :: preorder(btree *root)
{ btree *temp;
temp=root;
if(temp!=NULL)
{ cout<<temp->data;
preorder(temp->left);
preorder(temp->right);
}
}
void EXP_TREE :: nonrecpreorder(btree *root)
{ btree *current,
*s[10];
int top=-1;
if(root==NULL)
{
cout<<"\n The Tree is empty\n";
return; }
current=root;
for(;;) {
while(curren
t!=NULL)
{ cout<<""<<current->data;
push1(current,&top1,s);
current=current->left;
} if(!stempty1(top1)
{
Pop1(&top1,s,&current); current=current->right;
} else
return;
} }
void EXP_TREE :: postorder(btree *root)
{ btree
*temp;
temp=root;
if(temp!=NULL)
{ postorder(temp->left);
postorder(temp->right);
cout<<""<<temp->data;
}
}

void EXP_TREE :: nonrecpostorder(btree *root)
```

```
{ struct
stack
{

btree *element;
int check;
}st[10]; Int
top=-1; btree
*current;
if(root==NULL)
{
cout<<"\n The tree is empty";
return; }
current=root;
for(;;)
{
while(current!=NULL)
{ top++;
st[top].element=current;
st[top].check=1;
current=current->left;
}

while(st[top].check==0)
{
current=st[top].element;
top--; cout<<""<<current-
>data; if(stempty1(top))
return; }
current=st[top].element;
current=current->right;
st[top].check=0;
} }
void EXP_TREE :: push1(btree *item,int *top1,btree *s[])
{
*top1=*top1+1;
s[*top]=item; }
void EXP_TREE :: pop1(int *top,btree *s[],btree **current)
{
*current=s[(*top1)--]
}
int EXP_TREE :: stempty1(int top1)
{
If(top1==-1)
{ return 1;
else return
```

```
0; } int
main() {
char exp[80]; EXP_TREE
obj;
cout<<" Enter the postfix expression";
cin>>exp; obj.create(exp); cout<<"\n
The tree is created.\n"; cout<< "\n The
inorder traversal of it";
obj.inorder(obj.root);
cout<<"\n The non recursive inorder traversal of it";
obj.nonrecinorder(obj.root); cout<< "\n The preorder
traversal of it"; obj.preorder(obj.root); cout<<"\n The
non recursive preorder traversal of it";
obj.nonrecpreorder(obj.root); cout<< "\n The
postorder traversal of it"; obj.postorder(obj.root);
cout<<"\n The non recursive postorder traversal of
it"; obj.nonrecpostorder(obj.root);
return 0  }

//Out put:
Enter the postfix expression:
ab*c+
The tree is created; The
inorder traversal of it a *
b + c
The non recursive inorder traversal of it a
* b + c
The preorder traversal of it
+ *a b  c
The non recursive preorder traversal of it
+*abc
The postorder traversal of it a
b * c+
The non recursive postorder traversal of it a
b * c+
```

Assignment No 7

Title: Create Binary Tree (BT) and perform following operations:

a. Insert
b. Display
c. Depth of a tree
d. Display leaf-nodes
e. Create a copy of a tree

Program:

A) Program to create and display binary tree

```
#include<iostream.h>
#include<conio.h>
#include<process.h> struct
tree_node
{       tree_node *left;
tree_node *right;
        int data;
} ;
class bst
{       tree_node *root;
public:         bst()   {
root=NULL;
        }
        int isempty()
        {
                return(root==NULL);
        }
        void insert(int item);  void
inordertrav();  void inorder(tree_node
*);     void postordertrav();  void
postorder(tree_node *);         void
preordertrav();
        void preorder(tree_node *);
};
void bst::insert(int item)
{       tree_node *p=new tree_node;
tree_node *parent;      p->data=item;
p->left=NULL;           p-
>right=NULL;           parent=NULL;
if(isempty())
                root=p;
else
        {
                tree_node *ptr;
ptr=root;               while(ptr!=NULL)
                    {
```

```
                        parent=ptr;
if(item>ptr->data)
ptr=ptr->right;
                                else
                                        ptr=ptr->left;
                        }
                        if(item<parent->data)
                                parent->left=p;
                              else
                                parent->right=p;
        }
}
void bst::inordertrav()
{
        inorder(root);
}
void bst::inorder(tree_node *ptr)
{
        if(ptr!=NULL)
        {
              inorder(ptr->left);
cout<<"  "<<ptr->data<<"     ";
                inorder(ptr->right);
        }
}
void bst::postordertrav()
{
        postorder(root);
}
void bst::postorder(tree_node *ptr)
{
       if(ptr!=NULL)          {
postorder(ptr->left);
postorder(ptr->right);
                cout<<"  "<<ptr->data<<"     ";
        }
}
void bst::preordertrav()
{
        preorder(root);
}
void bst::preorder(tree_node *ptr)
{
        if(ptr!=NULL)
        {
              cout<<"  "<<ptr->data<<"     ";
preorder(ptr->left);              preorder(ptr->right);
```

```
        }
}



void main()
{
        bst b;
        b.insert(52);
        b.insert(25);
        b.insert(50);
        b.insert(15);
        b.insert(40);
        b.insert(45);
        b.insert(20); cout<<"inorder"<<endl;
      b.inordertrav();
cout<<endl<<"postorder"<<endl;    b.postordertrav();
cout<<endl<<"preorder"<<endl;     b.preordertrav();
        getch();
}

//output
inorder
 15    20    25    40    45    50    52 postorder
 20    15    45    40    50    25    52 preorder
 52    25    15    20    50    40    45
```

B) Program to create copy of a Binary tree

```
#include<iostream> #include<map>
using namespace std;

/* A binary tree node has data, pointer to left child, a pointer to right child and a pointer
to random node*/

struct Node
{   int
key;
  struct Node* left, *right, *random;
};

/* Helper function that allocates a new Node with the
given data and NULL left, right and random pointers. */
Node* newNode(int key)
{
  Node* temp = new Node;    temp-
>key = key;
```

```
    temp->random = temp->right = temp->left = NULL;
return (temp);
}

/* Given a binary tree, print its Nodes in inorder*/

void printInorder(Node* node)
{
    if (node == NULL)
        return;

    /* First recur on left sutree */

    printInorder(node->left);

    /* then print data of Node and its random
*/     cout << "[" << node->key << " ";     if
(node->random == NULL)
        cout << "NULL], ";
else
        cout << node->random->key << "], ";

    /* now recur on right subtree */     printInorder(node-
>right);
}

// This function creates clone by copying key and left and right pointers //
This function also stores mapping from given tree node to clone.

Node* copyLeftRightNode(Node* treeNode, map<Node *, Node **mymap)
{
    if (treeNode == NULL)
        return NULL;
    Node* cloneNode = newNode(treeNode->key);
    (*mymap)[treeNode] = cloneNode;
    cloneNode->left  = copyLeftRightNode(treeNode->left, mymap);
cloneNode->right = copyLeftRightNode(treeNode->right, mymap);
return cloneNode;
}

// This function copies random node by using the hashmap built by
// copyLeftRightNode()

void copyRandom(Node* treeNode,  Node* cloneNode, map<Node *, Node **
*mymap)
{
    if (cloneNode == NULL)
```

```
        return;
    cloneNode->random   =      (*mymap)[treeNode->random];
copyRandom(treeNode->left,     cloneNode->left,     mymap);
copyRandom(treeNode->right, cloneNode->right, mymap);
}


// This function makes the clone of given tree. It mainly uses
// copyLeftRightNode() and copyRandom()

Node* cloneTree(Node* tree)
{     if (tree ==
NULL)
        return NULL;
    map<Node *, Node *> *mymap = new  map<Node *, Node *>;
Node* newTree = copyLeftRightNode(tree, mymap);
copyRandom(tree, newTree, mymap);
    return newTree;
}


/* Driver program to test above functions*/

int main() {
    //Test No 1
    Node *tree = newNode(1);     tree-
>left = newNode(2);     tree->right =
newNode(3);     tree->left->left =
newNode(4);     tree->left->right =
newNode(5);     tree->random = tree-
>left->right;     tree->left->left-
>random = tree;
    tree->left->right->random = tree->right;

    //  Test No 2
    //    tree = NULL;

    //  Test No 3
    //    tree = newNode(1);

    //  Test No 4     /*    tree =
newNode(1);         tree->left =
newNode(2);         tree->right =
newNode(3);         tree->random
= tree->right;
        tree->left->random = tree;
    */
```

```
    cout << "Inorder traversal of original binary tree is: \n";
  printInorder(tree);

    Node *clone = cloneTree(tree);

    cout << "\n\nInorder traversal of cloned binary tree is: \n";
  printInorder(clone);

    return 0;
  }
```

```
//output
Inorder traversal of original binary tree is:
[4 1], [2 NULL], [5 3], [1 5], [3 NULL],

Inorder traversal of cloned binary tree is:
[4 1], [2 NULL], [5 3], [1 5], [3 NULL],
```

C . PROGRAM TO FIND DEPTH OF BINARY TREE

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

/* A binary tree node has data, pointer to left child
and a pointer to right child */

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

/* Compute the "10Depth" of a tree -- the number of
nodes along the longest path from the root node      down
to the farthest leaf node.*/

int 10Depth(struct node* node)
{    if
(node==NULL)
return 0;    else
   {
      /* compute the depth of each subtree */
int lDepth = 10Depth(node->left);
      int rDepth = 10Depth(node->right);
```

```
      /* use the larger one */
if (lDepth > rDepth)
return(lDepth+1);       else
return(rDepth+1);
  }
}

/* Helper function that allocates a new node with the
  given data and NULL left and right pointers. */

struct node* newNode(int data)
{    struct node* node = (struct
node*)    malloc(sizeof(struct
node));    node->data = data;
node->left = NULL;
   node->right = NULL;

   return(node);
}

int main() {
   struct node *root = newNode(1);

   root->left = newNode(2);    root-
>right = newNode(3);    root->left-
>left = newNode(4);
   root->left->right = newNode(5);

   cout<<"Hight of tree is "<<10Depth(root);

   getchar();
return 0;
}


//OUT PUT
Hight of tree is 3
```

D. PROGRAM TO PRINT LEAF NODES.

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

/* A binary tree node has data, pointer to left child
and a pointer to right child */
```

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


/* Function to get the count of leaf nodes in a binary tree*/

unsigned int getLeafCount(struct node* node)
{
 if(node == NULL)
   return 0;
 if(node->left == NULL && node->right==NULL)
   return 1;
 else
   return getLeafCount(node->left)+           getLeafCount(node-
>right);
}


/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */


struct node* newNode(int data)
{   struct node* node = (struct
node*)
                    malloc(sizeof(struct node));
  node->data = data;   node-
>left = NULL;
  node->right = NULL;

  return(node);
}
 /*Driver program to test above functions*/

int main()
{
  /*create a tree*/    struct node
*root = newNode(1);   root->left
= newNode(2);   root->right       =
newNode(3);   root->left->left  =
newNode(4);
  root->left->right = newNode(5);

  /*get leaf count of the above created tree*/
```

```
  COUT<<"Leaf count of the tree is"<<getLeafCount(root);
getchar();   return 0; }
//OUTPUT
Leaf count of the tree is  3
```

Assignment No 8

Title Program to implement Threaded Binary Tree

Program: A. Inorder threaded binary tree traversal.

```
#include <iostream>
#define 10_VALUE 65536
using namespace std; class
N { //node declaration
public:      int k;      N *l,
*r;
    bool leftTh, rightTh;
};
class ThreadedBinaryTree {
private:   N *root;   public:
  ThreadedBinaryTree() { //constructor to initialize the variables
root= new N();      root->r= root->l= root;      root->leftTh =
true;
    root->k = 10_VALUE;
  }
  void insert(int key) {
N *p = root;
    for (;;) {
    if (p->k< key) { //move to right thread
      if (p->rightTh)
break;        p = p-
>r;
    }
    else if (p->k > key) { // move to left thread
      if (p->leftTh)
break;        p = p-
>l;
    }
else {
return;
    }
  }
  N *temp = new N();   temp-
>k = key;
  temp->rightTh= temp->leftTh= true;
  if (p->k < key) {
temp->r = p->r;
temp->l= p;      p->r
= temp;
    p->rightTh= false;
  }
  else {      temp->r
= p;      temp->l = p-
```

```
>l;      p->l = temp;
p->leftTh = false;
  } }
void inorder() { //print the tree
N *temp = root, *p;    for (;;)
{      p = temp;      temp =
temp->r;      if (!p->rightTh)
{         while (!temp->leftTh)
{
          temp = temp->l;
        }
     }
     if (temp == root)
break;
        cout<<temp->k<<" ";
     }
     cout<<endl;
  }
};

int main()
 {
  ThreadedBinaryTree tbt;
cout<<"Threaded Binary Tree\n";
tbt.insert(56);    tbt.insert(23);
tbt.insert(89);    tbt.insert(85);
tbt.insert(20);    tbt.insert(30);
tbt.insert(12);    tbt.inorder();
cout<<"\n";
}
```

Assignment No 9

Program:

Program to implement prim‟s algorithm

```
#include<iostream>
#define size 20
#define 10 32767
using namespace std;
class mst { private: int
g[size][size],nodes;
public: mst(); void
prim(); void getdata();
}; mst::mst() { for(int
i=0;i<size;i++) for(int
j=0;j<size;j++)
g[i][j]=0; }
void mst::prim()
{ int select[size],i,j,k; int mindist,v1,v2,total=0;
for(i=0;i<nodes;i++) select[i]=0                    //initialize
the selected virtices list. cout<<"\n The minimum spanning
tree is";
select[0]=1;

for(k=1;k<nodes;k++)
{ mindist=10;
for(i=0;i<nodes;i++)
{
for(j=0;j<nodes;j++)
{ if(g[i][j]&&((select[i]&&!select[j]) ||
(!select[i]&&select[[j])))
{
if(g[i][j]<mindist)
{
mindist=g[i][j];
v1=i; v2=j;
}   } }   }
cout<<"\n Edge ("<<v1<<" "<<v2<<") and weight = "<<mindist; select[v1]=select[v2]=1;
total=total+mindist; } cout<<"\n total
path length is"<<total;
} void mst::getdata() { int v1,v2,length,n;
cout<<"\n enter the no of nodes in the tree";
cin>>nodes; cout<<"\n enter the no of
edges in the tree"; cin>>n; cout<<"\n enter
edges and weight"; for(int i=0;i<n;i++) {
cout<<"Enter edge by v1 and v2";
cin>>v1>>v2; cout<<"\n Enter
```

corresponding weight”; cin>>length;
g[v1][v2]=g[v2][v1]=length;
} } int main() { mst obj;
cout<<”\n prims algorithm”;
obj.getdata(); cout<<”\n\t”;
obj.prim(); return 0;
}
//OutPut
Enter the no of nodes in the tree 7
Enter the no of edges in the tree 9
Enter edges and weights
Enter edge by v1 and v2 : 0 1
Enter corresponding weight : 27
Enter edge by v1 and v2 : 1 2
Enter corresponding weight :16
Enter edge by v1 and v2 : 2 3
Enter corresponding weight :1 2
Enter edge by v1 and v2 : 3 4
Enter corresponding weight : 22
Enter edge by v1 and v2 : 4 5
Enter corresponding weight : 25
Enter edge by v1 and v2 : 0 5
Enter corresponding weight :6
Enter edge by v1 and v2 : 16
Enter corresponding weight :14
Enter edge by v1 and v2 : 4 6
Enter corresponding weight : 24
Enter edge by v1 and v2 : 3 6
Enter corresponding weight :18 The
minimum spanning tree is:
Edge(0 5)and weight=6
Edge(0 5)and weight=25
Edge(0 5)and weight=22
Edge(0 5)and weight=12
Edge(0 5)and weight=16
Edge(0 5)and weight=14 Total path length is: 95 Program to implement Kruskal‟s algorithm
#include<iostream>
#define 10 999 using
namespace std; class
kruskal {private:
typedef struct graph
{
int v1,v2,cost; }gr; gr
g[10]; public: int
tot_edges,tot_nodes;
void create(); void

```
spanning_tree(); void
getdata();
int minimum(int);
}; int find(int v2,int
parent[])
{while(parent[v2]!=v2)
{v2 =parent[v2];
} return
v2;}
void union(int i, int j, int parent[])
{if(i<j)
{parent[j]=i;
else parent[i]=j;
}

void kruskal::getdata()
{cout<<"\n Enter total number of nodes"; cin>>tot_nodes;
cout<<"\n Enter total number of edges"; cin>>tot_edges;
}
void kruskal::minimum(int n)
{int i,small,pos;
small=10; pos=-1;
for(i=0;i<n;i++)
{if(g[i].cost<small)
{small=g[i].cost;
pos=i; }} return
pos; }
void kruskal::spanning_tree()
{int count,k,v1,v2,i,j,tree[10][10],pos,parent[10];
int sum; count=0; sum=0;
for(i=0;i<tot_nodes;i++) parent[i]=i;
while(count!=tot_nodes-1;i++)
{pos=minimum(tot_edges); if(pos==-1) break;
v1=g[pos].v1; v2=g[pos].v2; i=find(v1,parent);
j=find(v2,parent); if(i!=j)
{tree[k][0]=v1;
tree[k][1]=v2; k++; count++;
sum+=g[pos].cost;
union(i,j,parent);}
g[pos].cost=10;}
if(count==tot_nodes-1)
{cout<<"\n spanning tree is";
for(i=0;i<tot_nodes-1;i++)
{cout<<"["<<tree[i][0]; cout<<" ";
cout<<tree[i][1]<<"]"<<endl;} cout<<"\n cost of
spanning tree is="<<sum<<endl;} else
{cout<<"There is no spanning tree";
```

```
} } int main() { Kruskal obj;
cout<<"\n \t Graph Creation ";
obj.getdata(); obj.create();
obj.spanning_tree();
return 0;
}
```

```
Graph Creation
Enter total number of nodes: 4
Enter total number of edges: 5
Enter edge in (v1 v2)  form 1 4
Enter corresponding cost 3
Enter edge in (v1 v2)  form 1 2
Enter corresponding cost 8
Enter edge in (v1 v2)  form 2 3
Enter corresponding cost 6
Enter edge in (v1 v2)  form 2 4
Enter corresponding cost 2
Enter edge in (v1 v2)  form 3 4
Enter corresponding cost 1 Spanning
tree is:
[3-4]
[2-4]
[1-4]
Cost of spanning tree is: 6
```

Assignment No. 10:

Title: Program to implement Dijkstra‟s algorithm using priority queue.

Program:

Program to implement Dijkstra‟s algorithm using priority queue.

```
#include<iostream>
#define member 1
#define nonmember 0
#define infinity 999
#define 10 10 using
namespace std;
class sp { private: int
g[10][10],q[10]; public: int
front,rear,n; void
build_graph(); void
dikstra(int,int); void
insert_q(int); int
delet_q(int);
};
void sp::build_graph()
{ int i,j,v1,v2;
for(i=1;i<=n;i++)
{
for(j=1;j<=1;j++)
{ cout<<”\n Enter the edge of V”<<i<<”to
V”<<j<<”:”; cin>>g[i][j];
} cout<<”\n”;
} } void sp::insert_q(int
index)
{ q[index]=1;  //node with smallest path is
inserted.
}
void sp:: delet_q(int i)
{
if(q[i]==1)  //smallest path length
{ return i; return infinity; //if it is not a smallest
path node.
}

void sp::dikstra(int src,int dest)
{
int small,dist[10],current,start,new1; int
temp,i;
for(i=0;i<=n;i++)
{ q[i]=0;
dist[i]=infinity;
```

```
} q[src]=1;                    //starting from source node
dist[src]=0;                   //initial distance is zero
current=src;          //consider source node as a current node
while(current!=dest)
{ small=infinity;
start=dist[current];
for(i=1;i<=n;i++)
{ if(q[i]==0) {
new1=start+g[current][i];
if(new1<dist[i])
dist[i]=new1;
if(dist[i]<small)              //finding smallest dist
{ small=dist[i];
temp=i;
}
}
} }
current=temp;
insert_q(current);
} } int
main() {
int scr,dest,path_node,k; sp
obj;
cout<<"\n Enter the number of vertices";
cin>>obj.n; obj.buld_graph(); cout<<"\n
Eneter the source"; cin>>src;
cout<<"\n Eneter the destination";
cin>>dest; obj.dikstra(src,dest);
cout<<"\n The sortest path is ";
obj.front=1; obj.rear=obj.n;
while(obj.front<=obj.rear)
{ path_node=obj.delet_q(obj.front);
if(path_node!=infinity)
cout<<"  "<<path_node;
obj.front++;
} return
0; }
#OutPut
Enter the number of vertices: 5 Enter
the edge of v1 to v1: 999
Enter the edge of v1 to v2:9
Enter the edge of v1 to v3:4
Enter the edge of v1 to v4:999
Enter the edge of v1 to v5:999
Enter the edge of v2 to v1:9
Enter the edge of v2 to v2:999
Enter the edge of v2 to v3:999
```

Enter the edge of v2 to v4:3
Enter the edge of v2 to v5:999
Enter the edge of v3 to v1:4
Enter the edge of v3 to v2:999
Enter the edge of v3 to v3:999
Enter the edge of v3 to v4:999
Enter the edge of v3 to v5:1
Enter the edge of v4 to v1:999
Enter the edge of v4 to v2:3
Enter the edge of v4 to v3:999
Enter the edge of v4 to v4:999
Enter the edge of v4 to v5:1
Enter the edge of v5 to v1:999
Enter the edge of v5 to v2:999
Enter the edge of v5 to v3:1
Enter the edge of v5 to v4:1
Enter the edge of v5 to v5:999
Enter the source: 1
Enter the destination: 5 The shortest path is: 1 3 5

Assignment No. 11

Title: Implement Heap Sort by constructing 10 or min heap.

Program:

Implement Heap Sort by constructing 10 or min heap.

```
#include<iostream>
#define 10 10 using
namespace std class
heap { private: int
arr[10]; int n;
public: heap(); void
insert(int num); void
makeheap(); void
heapsort(); void
display();
}; heap::heap() {
n=0;
for(i=0;i<10;i++)
a[i]=0; }
void heap::insert(int num)
{ if(n<10) {
arr[n]=num;
n++; } else
cout<<"\n Arrary is Full";
}
void heap::makeheap()
{
for(int i=1;i<n;i++)
{ int
val=arr[i]; int
j=i;
int f=(j-1)/2; while(j>0&&arr[f]<val)
{
arr[j]=arr[f];
j=f; f=(j-
1)/2; }
arr[j]=val;
}
} void heap::
heapsort() { for(int
i=n-1;i>0;i--) { int
temp=arr[i];
arr[i]=arr[0]; int k=0;
int j;
if(i==1) j=-1; else j=1;
if(i>2&&arr[2]>arr[1]) j=2;
while(j>=0;&&temp<arr[j]) {
```

```
arr[k]=arr[j]; k=j; j=2*k+1; if
(j+1<=i-1&&arr[j]<arr[j+1])
j++; if(j>i-1) j=-1; }
arr[k]=temp; } } void heap::
display() { for(int i=0;i<n;i++)
cout<<"\n"<<arr[i];
cout<<"\n"; } int main() {
heap obj; obj.insert(14);
obj.insert(12); obj.insert(9);
obj.insert(8); obj.insert(7);
obj.insert(10); obj.insert(18);
cout<<"\n The elements are:"
obj.display(); obj.makeheap();
cout<<"\n Heapifield"<<endl;
obj.display(); obj.heapsort();
return 0; }
//Output The
elements are:
14
12
9
8
7
10
18
Heapifield
18
12
14
8
7
9
10
Element sorted by heap sort:
7
8
9
10
12
14
18
```

Assignment No. 12:

Program:

Implementation of index sequential file for any database and perform following operation
1.Create 2.Display 3.Add record 4.Delete record 5. Modify record

```
#include<iostream>
#include<iomanip>
#include<fstream>
#include<cstring>
#include<stdlib.h> using
namespace std; class
emp_class
{
typedef struct emp
{ char
name[20];
int emp_id;
int sal; }rec;
typedef struct index
{ int emp_id; int
position; }ind_rec;
rec records; ind_rec
ind_records; public:
emp_class(); void
create(); void
display(); void
update(); void
delet(); void
append(); void
search();
};
emp_class::emp_class()
{ strcpy(records.name,"");
}
void emp_class::create()
{
int i,j;
char ch="y";
fstream seqfile;
fstream indexfile;
i=0;
indexfile.open("ind.dat",ios::in|ios::out|ios::binary);
seqfile.open("emp.dat", ios::in|ios::out|ios::binary);
do { cout<<"\n Enter name";
cin>>records.name;
cout<<"\nEnter emp_id";
cin>>records.emp_id; cout<<"\n
```

```
Enter salary";
cin>>records.salary;
seqfilewrite((char*)&records,sizeof(records))<<flush;
ind_records.emp_id=records.emp_id; ind_records.position=i;
indexfile.write((char*)&ind_records,sizeof(ind_records))<<flush;
i++; cout<<"\n Do u want to add more records"; cin>>ch;
}while(ch=="y"); seqfile.close(); indexfile.close();
}
void emp_class::display()
{ fstream
seqfile;
fstream indexfile; int n,i,j; seqfile.open("em.dat",
ios::in|ios::out|ios::binary); indexfile.open("ind.dat",
ios::in|ios::out|ios::binary);
indexfile.seekg(0,ios::beg);
seqfile.seekg(0,ios::beg); cout<<"\n The content of
file are"<<endl;
i=0;
while(indexfile.read((char *)&ind_records,sizeof(ind_records)))
{ i=ind_records.position*sizeof(rec);
seqfile.seekg(i,ios::beg);
seqfile.read((char *)&records,sizeof(records)); if(records.emp_id!=-1)
{
cout<<"\n Nmae"<<records.name<<flush;
cout<<"\n emp_id:"<<records.emp_id;
cout<<"\n Salary:"<<records.salary; cout<<"\n";
} }
seqfile.close();
indexfile.close();
}
void emp_class::update()
{
int pos,id;
char New_name[20]; int
New_emp_id; int
New_salary; cout<<"\n
for updation"; cout<<"\n
Enter the emp_id to
search"; cin>>id;
fstream seqfile; fstream
indexfile; int n,i,j;
seqfile.open("em.dat", ios::in|ios::out|ios::binary); indexfile.open("ind.dat",
ios::in|ios::out|ios::binary); indexfile.seekg(0,ios::beg);
pos=-1;
while(indexfile.read((char *)&ind_records,sizeof(ind_records)))
{
if(id==ind_records.emp_id)
```

```
{
pos=ind_records.position; break; } } if(pos==-
1) { cout<<"\nThe record is not present in the
file";
return; }else { cout<<"\n Enter the values
for updation"; cout<<"\n Eneter Name";
cin>>New_name;
cout<<"\n Enter salary";
cin>>New_salary; int
offset=pos*sizeof(rec);
seqfile.seekp(offset);
strcpy(records.name,New_name);
records.emp_id=id;
records.salary=New_salary;
seqfile.write((char*),&records,sizeof(records))<<flush; cout<<"\nThe
record is updated";
} seqfile.close();
indixfile.close(); }
void emp_class::delet()
{ int id,pos; cout<<"\n For deletion";
cout<<"\n Enter the emp_id to search";
cin>>id; fstream seqfile; fstream
indexfile;
seqfile.open("em.dat", ios::in|ios::out|ios::binary);
indexfile.open("ind.dat", ios::in|ios::out|ios::binary);
indexfile.seekg(0,ios::beg);
seqfile.seekg(0,ios::beg); pos=-1;
while(indexfile.read((char *)&ind_records,sizeof(ind_records)))
{
if(id==ind_records.emp_id)
{ pos=ind_records.position;
ind_records.emp_id=-1;
break; } } if(pos==-1) { cout<<"\nThe record is
not present in the file"; return; }
//Calculating the position of record in seq. fileusing the pos of index file.
int offset=pos*sizeof(rec); seqfile.seekp(offset);
strcpy(records.name,""); records.emp_id=-1; records.salary=-1;
seqfile.write((char*)&records,sizeof(records))<<flush;
offset=pos*sizeof(ind_rec); indexfile.seekp(offset);
ind_records.emp_id=-1; ind_records.position=pos;
indexfile.write((char*)&ind_records,sizeof(ind_records))<<flush;
seqfile.seekg(0); indexfile.close(); seqfile.close(); cout<<"\n The
record is deleted"; }

void emp_class::append()
{ fstream seqfile; fstream indexfile; int pos;
indexfile.open("ind.dat",ios::in|ios::binary);
```

```
indexfile.seekg(0,ios::end);
pos=indexfile.tellg()/sizeof(ind_records);
indexfile.close();
indexfile.open("ind.dat",ios::app|ios::binary);
seqfile.open("emp.dat",ios::app|ios::binary);
cout<<"\n Enter the record for appending";
cout<<"\n Name"; cin>>records.name;
cout<<"\n emp_id"; cin>>records.emp_id;
cout<<"\n Salary"; cin>>records.salary;
seqfile.write((char*)&records,sizeof(records)); ind_records.emp_id=records.emp_id;
ind_records.position=pos;
indexfile.write((char*)&ind_records,sizeof(ind_records))<<flush;
seqfile.close(); indexfile.close();
cout<<"\n The record is appended";
}
void emp_class::search()
{ fstream seqfile; fstream indexfile; int id,pos,offset;
cout<"\n Enter the emp_id for searching the redords";
cin>>id; indexfile.open("ind.dat",ios::in|ios::binary);
pos=-1;


while(indexfile.read((char *)&ind_records,sizeof(ind_records)))
{
if(id==ind_records.emp_id) {
pos=ind_records.position;
break; }
}

if(pos==-1) { cout<<"\nThe record is not
present in the file"; return;
}//calculating offset using position obtained from ind.dat
offset=pos*sizeof(records);
seqfile.open("emp.dat",ios::in|ios::binary);
seqfile.seekg(offset,ios::beg);
seqfile.read((char*)&records,sizeof(records)); if(records.emp_id==-1)
{ cout<<"\n Records is not present in the
file";
return; } else { cout<<"\n The record is present in
the file and it is:";
cout<<"\n Name:"<<records.name; cout<<"\n
emp_id"<<records.emp_id; cout<<"\n
salary:"<<records.salary;
} seqfile.close();
indexfile.close();
 }
```

```
int main() {
emp_class list;
char ans="y";
int ch,key; do
{
cout<<"\n 1.Create \n 2.Display \n 3.Update \n 4.Delete \n 5.Append \n 6.Search \n 7.Exit";
cout<<"\n Enter your choice"; cin>>ch; switch(ch)
{case 1:list.create();break;
case 2:list.display();break;
case 3:list.update();break;
case 4:list.delet();break;
case 5:list.append();break;
case 6:list.search();break;
case 7:break; }
}while(ans=="y");
return 0;  }
```

OutPut

```
1.Create
2.Display
3.Update
4.Delete
5.Append
6.Search
7.Exit
Enter your choice: 1
Enter name :abc
Enter emp_id: 10
Enter salary :1000
Do u want to add more record? n
1.Create
2.Display
3.Update
4.Delete
5.Append
6.Search
7.Exit
Enter your choice: 2
The content of file are
Name:abc
Emp_id:10
Salary: 1000

1.Create
2.Display
3.Update
4.Delete
5.Append
```

6.Search
7.Exit Enter your
choice: 3 For
updation:
Enter the emp_id to search 10
Enter the values for updation
Name: asd
Salary2000
The record is updated
1.Create
2.Display
3.Update
4.Delete
5.Append
6.Search
7.Exit
Enter your choice: 2
The content of file are
Name:asd
Emp_id:10
Salary: 2000

1.Create
2.Display
3.Update
4.Delete
5.Append
6.Search
7.Exit
Enter your choice: 4
For deletion
Enter emp_id for deletion of record10 The
record is deleted.
1.Create
2.Display
3.Update
4.Delete
5.Append
6.Search
7.Exit
Enter your choice: 2
The record is not preset in the file.
1.Create
2.Display
3.Update
4.Delete
5.Append

6.Search
7.Exit
Enter your choice: 5
Enter the record for appending
Name: asd
Emp_id:10 Salary:2000
The record is appended.
1.Create
2.Display
3.Update
4.Delete
5.Append
6.Search
7.Exit
Enter your choice: 6
Enter the emp_id for searching the record: 10 The
record is present in the file and it is:
Name:asd
Emp_id:10
Salary:2000
1.Create
2.Display
3.Update
4.Delete
5.Append
6.Search
7.Exit
Enter your choice: 7