(II) 1. Develop a payroll application to compute the salary for the employees in an organization. Class Employee has general information namely employee_id, empl oyee name, department_name, address, mobile number and email -id. There are two types of employees such as training professionals and regular em ployees. The Definition of Pay_calculate() will be different for different types of e mployees. Include suitable menu for navigating the application.

Training professionals has total teaching experience, qualification, no_of_sessions _handled.
Salary =no_of_sessions_handled*one_day_salary (Both can be randomized values)

Regular employees has department they belong to, basic_pay, designation
Gross Pay = basic_pay+HRA+DA
Net Pay = Gross Pay PF
HRA = 9% of Basic Pay, DA = 90% of Basic Pay, PF = 20% of Basic Pay

SNo	Designation	Basic_Pay
1	Manager	30000
2	Technical Lead	25000
3	Senior Software Engineer	20000
4	Software Engineer	15000

•Get and display details of different types of employees using operator overloadin g(<< , >>) (20)
• Write suitable exceptions for the given scenario . (10)
•Write a friend function LiseDeptWise() to list all employees for a given department_name. (20)

```
//employee details using operator overloading and heirarchial inheritance
#include<iostream> using
namespace std;
#include<string.h>
int n,k;
class employee    //base class
{ protected:         //to inherit the base class
        int emp_id;    char
        addr[10],email[50]; public:
                char emp_name[10],dep_name[4];
                friend istream& operator>>(istream& input,employee& e)   //input
operator overloading
```

```
                {

        input>>e.emp_name>>e.emp_id>>e.dep_name>>e.addr>>e.email;
                        return input;
                }
                friend ostream& operator <<(ostream& output,employee& e)
//output operator overloading
                {

        output<<e.emp_name<<"\t"<<e.emp_id<<"\t"<<e.dep_name<<"\t"<<e.addr
<<"\t"<<e.email<<"\n";
             }
};
class regular_emp;
class training_prof : public employee      //derived class 1
{
        private:
                int teaching_exp,no_of_sessions_handled,one_day_salary;
               char qualification[10];
       public:
                int salary;
                void getinfo()
                {
  cout<<"enter the teaching experience,no of sessions handled and qualification\n";
                        cin>>teaching_exp>>no_of_sessions_handled>>qualification;
                }
                void pay_calc()
                {
                        cout<<"enter the one day salary\n";
                        cin>>one_day_salary;
                        try                              //exception handling
                        {
                                if(one_day_salary==0)
                                   throw(one_day_salary);
                                else
                                {
                                        salary=no_of_sessions_handled*one_day_salary;
                                }
```

```
                    }
                    catch(int k)
                    {
                            cout<<"exception caught: one day salary cannot be zero";
                    }
            }
            friend void listdept(training_prof,regular_emp);
};
class regular_emp : public employee        //derived class 2
{
        private:
              int basic_pay;               char
designation[10];          float
gross_pay,hra,da,pf,net_pay;   public:
      void pay_calc1()
            {
              int ch;
              do
                    {
                            cout<<"1.MANAGER \n 2.TECHNICAL LEAD \n
3.SENIOR SOFTWARE ENGINEER \n 4.SOFTWARE ENGINEER \n 5.EXIT";
                            cout<<"\n choose the designation\t";
                          cin>>ch;
              switch(ch)
                            {
                                    case 1:
                                            basic_pay=30000;
                                     hra=0.09*basic_pay;
               da=0.9*basic_pay;
pf=0.2*basic_pay;
gross_pay=basic_pay+hra+da;
net_pay=gross_pay-pf;
cout<<"netpay="<<net_pay<<"\n";
                                     break;
                          case 2:
                                            basic_pay=25000;
                                  hra=0.09*basic_pay;
            da=0.9*basic_pay;                      pf=0.2*basic_pay;
```

```
                                            gross_pay=basic_pay+hra+da;
                                      net_pay=gross_pay-pf;
                                              cout<<"netpay="<<net_pay<<"\n";
                                            break;
                                   case 3:
                                               basic_pay=20000;
                                   hra=0.09*basic_pay;
                 da=0.9*basic_pay;
          pf=0.2*basic_pay;
          gross_pay=basic_pay+hra+da;
          net_pay=gross_pay-pf;
                                              cout<<"netpay="<<net_pay<<"\n";
                                            break;
                                   case 4:
                                               basic_pay=15000;
                                   hra=0.09*basic_pay;
                 da=0.9*basic_pay;
          pf=0.2*basic_pay;
          gross_pay=basic_pay+hra+da;
          net_pay=gross_pay-pf;
                                              cout<<"netpay="<<net_pay<<"\n";
                                            break;
                                   default:
                                                 cout<<"enter the valid designation\n";
                                    }
                              }while(ch<5);

                  }
                  friend void listdept(training_prof,regular_emp);
          };
          void listdept(training_prof t[],regular_emp r[])
          {
                char deptname[4];
                int i,j;
                cout<<"Enter the department to be listed:";
                cin>>deptname; for(i=0;i<k;i++)
                { if (strcmpi(t[i].dep_name,deptname)==0)
                      {
                            cout<<t[i].emp_name<<endl;
                }
        }
        for(j=0;j<n;j++)
        {
                if(strcmpi(r[j].dep_name,deptname)==0)
                {
                        cout<<r[j].emp_name<<endl;
                }
        }
} int
main()
{
 training_prof          t[50];
regular_emp   r[50];     int
m,i,j,ch;
        do
        {
              cout<<"1.TRAINING PROFESSIONALS \n 2.REGULAR
EMPLOYEE\n 3.EXIT";                  cout<<"\nenter your choice\t";
                cin>>ch;
                switch(ch)
                {
                        case 1:
                                cout<<"enter the no of employess\t";
                                cin>>k;
                                for(i=0;i<k;i++)
                                {
                                      cout<<"enter the name,id,department
name,address,email\t";
                                        cin>>t[i];
                                }
                                for(i=0;i<k;i++)
                                {
                                        t[i].getinfo();
                                        t[i].pay_calc();
                                }
                                cout<<"NAME \t id  \t department \t address \t email \t
salary"<<"\n";
```

```
                                for(i=0;i<k;i++)
                                {
                                        cout<<t[i];
                                        cout<<t[i].salary<<"\n";
                                }
                               break;
                        case 2:
                                cout<<"enter the no of employees\t";
                                cin>>n;
                                for(j=0;j<n;j++)
                                {
    cout<<"enter the name,id,department name,address,email\t";
                                        cin>>r[j];
                                }
                                for(j=0;j<n;j++)
                               {

                                        r[j].pay_calc1();
                                }
                                      cout<<"NAME \t id \t department \t address \t email
"<<"\n";
                                for(j=0;j<n;j++)
                                {
                                        cout<<r[j];
                                }
   break;    case 3:     goto x;
                                break;
                }
        }while(ch<4);
        x:
          listdept(t,r);
          return 0;
}
```

III 1. Develop a banking application to computerize the operations in the bank. Acc ount holder has general information that they require to provide are account_id, acct_holder_name, address, mobile number and emailid. There are two type s of account holders such as savings account & current account. Interest calculatio n definition will be different for different types of account holders. Include suitable menu for navigating the application.

Savings account holder has purpose_of_account & balance Current savings account holder has type_of_business & balance Interest calculation will be done on the current balance.

Interest calculation for savings account

Interest = balance*8% of balance

Interest calculation for current account

Interest = balance*12% of balance

•Get and display details of different categories of account holders using operator overloading(<< , >>) (20)

• Write suitable exceptions for the given scenario . (10)

•Implement the interest_calc method in two different types of account holders an d invoke them. (Virtual Function) (20)

```
#include<iostream>
using namespace std;
#include<stdlib.h> class
banking
{
public:
        int acc_id;
        unsigned long  int  phno;
        char name[15],address[20],mail_id[20];
        friend istream & operator >> (istream & in,banking & b)
        {
                cout<<"enter acc_id,name,address,email_id,phno"<<endl;
            in>>b.acc_id>>b.name>>b.address>>b.mail_id>>b.phno;
return in;
         }
       friend ostream & operator << (ostream & out,banking & b)
        {
               out<<”acc_id:”<<b.acc_id<<endl<<”name:”<<b.name<<endl;
out<<”address:”<<b.address<<endl;
                out<<"email_id:"<<b.mail_id<<endl<<"phno:"<<b.phno<<endl;
return out;
         }
        virtual void interest_calc()=0;
};
```

```
Class  saving :  public banking
{    public:     int
bal, interest;
char purpose[20];
    void get1()
           {
               try
               {
                          cout<<"enter balance and purpose of interest:";
                          cin>> bal >>purpose;
                             if(bal<500)
                         throw(1);
               }
             catch(int i)
             {
                        cout<<"exception caught:balance cannot be less than 500"<<endl;
exit(0);
                }
           }
             void interest_calc()
            {
                 interest=bal*(0.08)*bal;
               cout<<"interest:"<<interest<<endl;
           } };
class current:public banking
{
public:
              int interest1,bal1;
           char type[20];
           void get2()
              {
                         cout<<"enter balance and type of business"<<endl;
                      cin>>bal1>>type;
           }
           void interest_calc()
              {
                      interest1=bal1*(0.12)*bal1;
         cout<<"interest:"<<interest1;
```

```
        }
}; int main() {
int ch,se;
 banking*b;
current c;  saving
s;
        cout<<"enter type of account"<<endl;
      cout<<"1.savings account"<<endl<<"2.current account"<<endl;
      cin>>ch;
        do
        {
           switch(ch)
                {
                 case 1:
                cin>>s;
                s.get1();
                cout<<s;
                b=&s;
                        b->interest_calc();
                        break;
case 2:
                        cin>>c;
        c.get2();
cout<<c;                    b=&c;
                        b->interest_calc();
                break;
                }
              cout<<"\ndo u want  to continue (Y/N)(1/0)"<<endl;
      cin>>se;
        }while(se!=0&&se==1);
}
```

(IV) 1.Create class matrix with the data members 2D array, row and column of type integer. **Overload the following operators** and write the operator functions necessarily as member or friend of the class matrix.

>>(extraction operator) – to read the values for the data members (10)

<< (insertion operator) – to display the values of the data members (10)
+ (addition operator) -- to add two matrices (10)
-(unary minus) – to negate the values of a matrix (10)
++(pre and post increment operator)
– to increment the values of the data members of a matrix (10)

Write the main() program to demonstrate the matrix object for unary and binary operator overloading.

SOLN:

```
#include<iostream>

Using namespace std;

Class matrix

{

     int m[20][20],r,c;

public:

    friend istream& operator >>(istream& in,matrix &m1)

   {

      Cout<<"\n\tEnter the dimensions:";

      Cin>>m1.r>>m1.c;

      Cout<<"\n\tEnter the 2D array:";

for(int i=0;i<m1.r;i++)           for(int

j=0;j<m1.c;j++)

in>>m1.m[i][j];

  }

  friend ostream & operator <<(ostream& out ,matrix &m1)

 {
```

```
          Cout<<”\n\tThe matrix is:”;
        for(int i=0;i<m1.r;i++)
        {
            Cout<<”\n”;
for(int j=0;j<m1.c;j++)
            {
                Cout<<m1.m[i][j]<<”      “;
             }
         }
      void operator –()
      {
           for(int i=0;i<m1.r;i++)
for(int j=0;j<m1.c;j++)
m[i][j]= -1*m[i][j];
      }
      matrix  operator +(matrix m1)
      {              matrix m3;              m3.r=r;
m3.c=c;              for(int i=0;i<r;i++)
for(j=0;j<c;j++)
m3.m[i][j]=m[i][j]+m1.m[i][j];         return
m3;
   }
    void operator ++()
```

```
  {         for(int i=0;i<r;i++)
for(int j=0;j<c;j++)
m[i][j]= ++ m[i][j];
    } }; int main() {         matrix
m1,m2,m3;        cout<<"\n\tEnter
1st  matrix  :";             cin>>m1;
cout<<"\n\tEnter   2nd   matrix:";
cin>>m2;            cout<<"\n\tFirst
matrix:";                    cout<<m1;
cout<<"\n\tSecond         matrix:";
cout<<m2;     m3=m1+m2;
   -m1;
     Cout<<"\n\t Added matrix:";
      Cout<<m3;
      Cout<<"\n\t(-m1) is :";      Cout<<m1;
     ++m2;
     Cout<<"\n\t(++m2) is  :";
     Cout<<m2;
     Return 0;
}
```

(V)Design a class Arrayas a template class with asingledimensionalarrayand sizeofthe arrayas data members. Template class need to work for different types of input (integer, float, char, char array, string, student, employee). Include the following member functions

	getArray()and printArray()to read and print the elements of the array	
-	sort()the array elements using bubblesortalgorithm.	
-	To enable sorting employee or student objects based on a data member.	

- SOLUTION:

```
#include<conio.h>

#include<iostream.h>

#define N 10 template

<class t> class array

{ t a[N]; int i,j; public:

void getarray() {

cout<<"\nEnter the array:";

for(i=0;i<N;i++)

cin>>a[i]; } void

printarray() {

cout<<"\nThe array is:";

for(i=0;i<N;i++)

cout<<a[i]<<" "; } void

sort() { for(i=0;i<N;i++) {

for(j=0;j<N-1-i;j++) {
```

```
if(a[j]>a[j+1]) { t
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
} }
};

void main() { clrscr();
cout<<"\n For integer type!!";
array <int> b;  b.getarray();
b.printarray();
b.sort();
b.printarray();
cout<<"\n For float
type!!";  array
<float> x;
x.getarray();
x.printarray();
x.sort();
x.printarray(); cout<<"\n
For char array type!!";
array <char> y;

 y.getarray();
y.printarray();
```

```
y.sort();

y.printarray();

getch(); }
```

(VI) 1. Create a class student with the data members roll number, name, marks arra y to hold marks for 5 subjects, total and a Boolean variable pass. Keep the followi ng member functions in the class.

Void Read() – To read the data members

Void Print() – To print the data members Void

Result()-Computes the total and pass.

Pass variable will be "true" if all the marks are >=50, otherwise, it is "false". (20)

Write a function template to sort the input values of type integer, float, char array, string and Student objects. Student objects need to be sorted based on their total.

• To sort integer, float and string values. (10)

• To sort strings made up of character array. (10)

• To sort student object based on the total. (10)

```
#include<iostream>

#include <stdlib.h>

using namespace std;

class stud  { public:

int rno,m1,m2,m3,m4,m5,tot;

char name[20];   float avg;
```

```
public:  void read()  {
cout<<"\n\nEnter the rno:";
cin>>rno;  cout<<"Enter the
name:";  cin>>name;
cout<<"Enter the first marks:";
cin>>m1;  cout<<"Enter the
second mark:";  cin>>m2;
cout<<"Enter the third mark:";
cin>>m3;  cout<<"Enter the
fourth mark:"; cin>>m4;
cout<<"Enter the fifth mark:";
cin>>m5;
tot=m1+m2+m3+m4+m5;
avg=tot/5; cout<<tot;
}   void
print()  {

if(m1>=50&&m2>=50&&m3>=50&&m4>=50&&m5>=50)
cout<<"The stud is"<<"PASS"<<endl;
else
cout<<"The stud is"<<"FAIL"<<endl;
}  void
result()
{
```

```
cout<<"The RNO:"<<rno<<endl; cout<<"The
NAME is:"<<name<<endl; cout<<"The
MARK1:"<<m1<<endl; cout<<"The
MARK2:"<<m2<<endl; cout<<"The
MARK3:"<<m3<<endl; cout<<"THE
MARK4:"<<m4<<endl; cout<<"THE
MARK5:"<<m5<<endl; cout<<"The TOTAL
IS :"<<tot<<endl; cout<<"THE AVERAGE
IS:"<<avg;
} };  int main()
{ stud
s[60],temp;
int i,j,n;
cout<<"THE NUMBER OF STUDENTS:";
cin>>n;
for(i=0;i<=n;i++)
s[i].read();
for(i=0;i<=n;i++)
s[i].print();
for(i=0;i<=n;i++) {
for(j=i+1;j<=n;j++)
{
if(s[i].tot<s[j].tot)
{ temp=s[i];

s[i]=s[j];

s[j]=temp;

}

} }

for(i=0;i<=n;i++)

{ s[i].result(); }

return 0;


}
```

(VI) .2. A stack is a linear list of elements for which all insertions and deletions (us ually accesses) are made at only one end of the list. Create a class with the data me mbers integer array and the size. Write the following member functions and write menu driven () program to use those functions.

• Push (int X): pushes an element X on to the top of the stack –
This function uses the IsFull() member function while pushing the data. (10)

•int Pop(): pops the last pushed element. This function uses the IsEmpty() member function while popping the data from the stack . (10)

•void Display(): to display the elements in the stack. (10)

Write a program to find whether the given string is palindrome or not. (Use stack c lass). (20)

```
#include<iostream>

#define size 5

#include<stdlib.h> using
```

```
namespace std; class
stack
{       int top,a[size],i;
        public:
                    stack()
                    {
                              top=-1;
                    }
                    int isempty()
                    {
                              if (top==-1)
                              {
                                        return 1;
                            }
                  else
                              {
                                        return 0;
                              }
                    }
                    int isfull()
                    {       if(top==size-1)
                              {
                                        return 1;
                              }
                              else
                              {
                                        return 0;
                              }
                    }
                    void push(int x)
                    {
                              if (!isfull())
                              {
                                    top++;
                            a[top]=x;
                              }
                              else
                              {
                                        cout<<"Overflow...";
                              }
                    }
                    int pop()
                    {
                              if(!isempty())
                              {
                                        return (a[top--]);
                                        }
                                        else
                                        {
                                                cout<<"Underflow...";
                                        }
                              }
                              void Display()
                              {
                                        for(i=0;i<=top;i++)
                                        {
                                                cout<<a[i]<<endl;
                                        }
                              }
                              int menu()
                              {
                                        int ch;
                                    cout<<"Menu\n1.Push\n2.Pop\n3.Display\n4.Exit\nEnter your
                    choice...";
                                        cin>>ch;
                                        return (ch);
                              }
                    }; int
                    main()
                    {
                         stack    s;        int
                        c=0,e,x;
                        while(c<5)
                                {
                                        c=s.menu();
                              switch(c)
                                        {
                                                case 1:        cout<<"Enter the element...";
                                                        cin>>e;
                                                      s.push(e);
                                                break;
                                                case 2:        x=s.pop();
                              cout<<"Popped element is "<<x;
                                                        break;
                                                case 3:        s.Display();
                                                        break;
                                                case 4:        exit (0);
                                        }
                                } return
                                0;
                    }
```

7.CREATE A CLASS COMPLEX WITH DATA MEMBERS REAL AND IMAGINARY OF INTEGER TYPE .READ TWO COMPLEX NUMBERS AND STORE THEM IN FILE.WRITE THE FOLLOWING MEMBER FUNCTIONS.

***READ AND PRINT THE COMPLEX NUMBERS USING STREAM OPERATORS(<<,>>)**

***OPERATOR +() TO PERFORM COMPLEX NUMBER ADDITION.**

***OPEN THE FILE,READ COMPLEX NUMBERS USING INPUT FILE STREAM AND PERFORM COMPLEX NUMBER ADDITION.**

***STORE THE RESULTANT COMPONENTS IN ANOTHER FILE(USING BINARY FILE STREAM)**

```
#include<iostream.h>

#include<conio.h>

#include<fstream.h>

class complex

{

 public:

  int real,img;

  friend istream &operator>>(istream &input,complex &c)             //CONDITION 1

 {

  cout<<"enter real and imag part"<<endl;

  input>>c.real>>c.img;

  return input;

 }

 friend ostream &operator<<(ostream &output,complex &c)                //CONDITION 1

 {

  output <<c.real<<"+j"<<c.img<<endl;

  return output;

 }


 complex operator +(complex c)  //CONDITION 2
 {
  complex temp;
  temp.real=real+c.real;
  temp.img=img+c.img;
  return temp;
}};
void main()
{
 complex c1,c2,c3,c4,c5;
 clrscr();
 cin>>c1;
 ofstream fout1;
 fout1.open("num1.txt");
 fout1.write((char*)&c1,sizeof(c1));
 fout1.close();
 cin>>c2;
 ofstream fout2;
 fout2.open("num2.txt");
 fout2.write((char*)&c2,sizeof(c2));
 fout2.close();
 c3=c1+c2;
 cout<<c3;
 cout<<endl<<"using file \n";
 ifstream fin1,fin2;
 fin1.open("num1.txt");
 fin2.open("num2.txt");
 fin1.read((char*)&c4,sizeof(c4));
 fin2.read((char*)&c5,sizeof(c5));
 c3.real=c4.real+c5.real;

 c3.img=c4.img+c5.img;

 cout<<c3;

 getch();

 }
```

/* (VIII) 1. Create a class student with the data members roll number, name, marks array to hold the three UT marks for the subject PDS-II, attendance (out of 5) and internal marks. Keep the following member functions in the class. ?
* To read and display the data members of student class. (10) ?
* Compute_Internal() Computes the internal marks. (10) ?
* Create a binary file student.dat and insert 10 student objects in it. (20) ?
* Find() To find the student record given the position of it. (20) ? */

CODE:

```
#include<iostream.h>
#include<fstream.h>
#include<conio.h> class
student
{  char name[25];  int
rno,marks[3],atten;
public:  int intern;

 void read()
 {
  cout<<"\nEnter rno,name,three marks for pds-2 UT's &Attendace(out of 5)\n";
cin>>rno>>name>>marks[0]>>marks[1]>>marks[2]>>atten;
 }

void compute_internals()
{
 //for computing internals 90%of avg marks +10%of //attendace is taken
float mavg;
```

```
 mavg=((marks[0]+marks[1]+marks[2])*90)/300;
float aavg;  aavg=(atten*10)/5;
intern=mavg+aavg;
}

void disp()
{
 cout<<"\nrno="<<rno<<"\nname="<<name<<"\nInternals for pds-2="<<intern; }

void Find()
{  ifstream
file;
 file.open("File.dat",ios::binary|ios::in);
 student temp;
int n;
 cout<<"\nEnter the nth record to be find";
cin>>n;
 int i;
 //to place the file pointer begining of the record to be found
file.seekg((n-1)*sizeof(temp),ios::beg);
file.read((char*)&temp,sizeof(temp));  temp.disp();
//write function to write a class data members in file void
Write()
{  student temp;
temp.read();
temp.compute_internals();
 ofstream file;
 file.open("File.dat",ios::app|ios::binary|ios::out);
file.write((char*)&temp,sizeof(temp));  file.close();
} }; void
main()
{
 //existing file contains old record,so file should be truncated
ofstream file;
 file.open("File.dat",ios::binary|ios::trunc|ios::out);
file.close();  clrscr();  student s;
 int n,i;
 cout<<"Enter no of students ";
cin>>n;  for(i=0;i<n;i++)
 {
  s.Write();
 }
 s.Find(); getch();
}
```

(IX)1. A Queue is a linear list of elements where an element will be inserted at the back end and deleted at the front end. Create a class queue with the data members integer array, front and rear. Include the following member functions in the class queue and write a menu driven (1. Insert, 2. Delete, 3. Display, 4. Exit) program to use the class queue.   Write suitable exceptions for the given scenario.   (10)

• bool IsEmpty(): return whether the queue is empty or not   (10)

• bool IsFull(): return whether the queue is full or not.   (10)

• Insert (int X): inserts an element X into the back end of the queue –
 This function uses the IsFull() member function while inserting the data.   (10)
•int Delete(): removes the element from the front end. This function uses the IsE mpty() member function while deleting the data from the queue.   (10)

• void Display(): to display the elements in the queue.   (10)

```
  #include<iostream> using namespace std; #define size 5 class queue

{

       int i,a[size],front,rear;

       public:

                 queue()

                 {

                         front=rear=-1;

                 }
```

```
bool isempty()
{
      if (front==-1)
return 1;
        else
          return 0;
}
bool isfull()
{
      if (rear==size-1)
        return 1;
      else
          return 0;
}
void insert(int x)
{
        if(isfull())
        {
                cout<<"Overflow..."<<endl;
        }
        else
        {
```

```
                  rear++;
              a[rear]=x;
          if (front=-1)
front=0;

               }
           } int
           delet()
           {
               if (isempty())
               {
                   cout<<"Underflow..."<<endl;
               }
               else
               {
                   if (front==rear)
                   {
                       cout<<"Deleted is "<<a[front++]<<endl;
                       front=rear=-1;
                   }
                   else
                   {
                       cout<<"Deleted is "<<a[front++]<<endl;
                   }
```

```
            }
        }
        void display()
        {
            if (isempty())
            {
                cout<<"Queue is empty"<<endl;
            }
            else
            {
                for(i=front;i<=rear;i++)
                {
                    cout<<a[i]<<endl;
              }
          }
        }
        int menu()
        {
            int ch;
           cout<<"1.Insert\n2.Delete\n3.Display\n4.Exit\nEnter your
choice...";
           cin>>ch;
       return ch;
        }
```

```
}; int
main() {

 queue   q;      int
c=0,e,x;
while(c<5)
          {
                 c=q.menu();
        switch(c)
                    {
                              case 1:        cout<<"Enter the element:";
                                             cin>>e;
                                             q.insert(e);
                                             break;
                              case 2:        q.delet();
                                             break;
                              case 3:        q.display();
                                             break;
                              case 4: exit(0);
                    }
          }
          return 0;
}
```

(XI) 1. Consider an example of declaring the examination result. Design three class es: Student, Exam and Result. The student class has data members such as roll nu mber, name and address. Create the class Exam by inheriting the Student class. Th e Exam class adds data members to representing the marks scored in six subjects. Derive the Result class from the Exam class and it has its own data member 'result ' which is of type bool. Include getResult() in Result class to set the corresponding value (true or false) based on the marks of the students given in Exam class.

Write an interactive program to model this relationship   (20)

Write a friend class to maintain the collection of result objects and include a memb er function to list the student details such as roll number, name, address, marks for six subjects based upon the result. (30)

```
#include <iostream>

using namespace std;

#include<string.h>  class

student

 {

protected:

              int rollno;

char name[25];

char address[40];

public:               void

getstudent()

             {

                   cout<<"enter the name,rollno,address"<<endl;

cin >> rollno >> name >> address;
```

```
                }
};
 class exam : public student
{
protected:
                int mark[6];
public:                 void
getexam()
                {
                        cout<<"enter the marks"<<endl;
for(int i=0; i<6; i++)
                       {
                               cin >> mark[i];
                        }
                    }
};
 class result : public exam
{
protected:
               bool result;
public:                bool
getresult()
                {
                        for(int i=0; i<6; i++)
if(mark[i] < 50)
return false;                          return true;
                  }
void process()
                  {
                           result = getresult();
if(result)                                    cout
<< "Pass";                              else
cout << "Fail";
                     }
friend class  show;
};  class show  {     public:
void display(result &r1)
             {
                       cout<<"rollno"<<r1.rollno;
cout<<"name"<<r1.name;
cout<<"address"<<r1.address;                         for(int i=0;i<6;i++)

                     {
                                cout<<"marks"<<r1.mark[i];
                      }
              }
   };
```

```
 main()
 {
             result r1;
show s;
r1.getstudent();
r1.getexam();
r1.process();
           s.display(r1);
} input:  101 dinesh
Chennai
 23
45
 67
 78
23  90
output:
enter the name,rollno,address
 enter the marks
Failrollno101namedineshaddresschennaimark23mark45mark67markenter the
name,rollno,address  enter the marks
Failrollno101namedineshaddresschennaimarks23marks45marks67marks78mar
ks23marks90mark23mark90
```

(XII) 1. Create a class called Circle with radius as data member and getCircle() and calcArea() as member functions to read the radius and calculate area of the circle. (10)

Create a class called Cone which is derived from the Circle class. Utilize the data members and the member functions of the base class by the derived class to find the volume ((1/3)*3.14*r*r*h) of a cone.

Create another class called Cylinder which is derived from the Circle class. Utilize the data members and the member functions of the base class by the derived class to find the volume (3.14*r*r*h) of a cylinder. (10)

- Get and retrieve the data members of cone and cylinder using operator overloading (20)

- Use virtual function to find volume of the cone and cylinder and invoke them (10)

```
#include<iostream> using namespace std; class circle

{

    public:

        float area,radius;

    void getcircle()

        {

            cout<<"Enter the radius :";

    cin>>radius;

        }

        virtual void calcarea()

        {

            area=3.14*radius*radius;

    cout<<"\nThe area of circle is:"<<area;

        }
```

```
};
class cone:public circle
{
    float volume,height;
    public:
            friend istream &operator >>(istream &input,cone &a)
            {
                cout<<"\nEnter the height of the cone:";
                input>>a.height;                return
input;

            }
            void calcarea()
            {
                volume=(0.33*3.14*radius*radius*height);
    cout<<"\nThe calculated volume of cone is:"<<volume;
            }
};
class cylinder:public circle
{
    float volume,height;
    public:
            friend istream &operator >>(istream &input,cylinder &a)
            {
```

```
                         cout<<"\nEnter the height of the cylinder:";
                        input>>a.height;
                   return input;
                    }
                    void calcarea()
                    {
                         volume=(3.14*radius*radius*height);
                         cout<<"\nThe calculated volume of cylinder is:"<<volume;
                    }
};
int main()
{
        cone c;
        cylinder y;
        circle *b,*d;
        b=&c;
        d=&y;
        cout<<"\n**cone**"<<endl;
        b->getcircle();       cin>>c;
        b->calcarea();
        cout<<"\n\n**cylinder**"<<endl;
        d->getcircle();       cin>>y;
          d->calcarea();
```

```
}
```

(XIII) 1. Create a class matrix with the data members **m, row and column of type integer. Write the following member functions

Overload stream operators (<<, >>) to read and display the matrix (20)

Overload binary minus (-) operator to subtract one matrix from the other (10)

Overload unary minus (-
) operator to subtract the values of the elements of a given matrix (10)

Allocate memory for the data member ‘m’ using constructor (10) Deallocate

the memory for ‘m’ using destructor (10)

```
//MATRIX
#include<iostream.h>
#include<conio.h>
 #include<stdlib.h>

 class matrix

 {
 private: int m[3][3];

 public :

friend istream &operator >>(istream &input,matrix &d)
 {
```

```
int i,j; for(i=0;i<3;i++) for(j=0;j<3;j++) input>>d.m[i][j]; return input;
  }


friend ostream &operator <<(ostream &output,matrix &d)
  {
  int i,j; for(i=0;i<3;i++)
  {
for(j=0;j<3;j++)
  {
output<<d.m[i][j]<<" ";
  }
cout<<"  \n";
  }
return output;
  }


matrix operator +(matrix &m2)
  {
  matrix temp; for(int i=0;i<3;i++)
for(int j=0;j<3;j++)
  {
temp.m[i][j]= m[i][j]+m2.m[i][j];
  }
return temp;
  }

matrix operator   -(matrix &m2)
  {
```

```
matrix temp; for(int i=0;i<3;i++)
for(int j=0;j<3;j++)
  {
temp.m[i][j]=m[i][j]   - m2.m[i][j];
  }
  return temp;
  }


~matrix()
  {
  }
  };

void main()
  {
  matrix m1,m2,m3,m4; cin>>m1;
  cin>>m2;    m3=m1+m2;
  m4=m1      -m2;
  cout<<"\nThe 1st  matrix is:"<<m1; cout<<"\nThe
  2nd  matrix is:"<<m2; cout<<"\nThe 3rd  matrix
  is:"<<m3; cout<<"\nThe 4th  matrix is:"<<m4;
  }
```

XV CREATE A CLASS TIME TO DISPLAY A TIME VALUE WITH DATA MEMBER OF HOURS, MINUTES AND SECONDS.INCLUDE THE FOLLOWINGMEMBERS FUNCTION TO GET AND DISPLAYTHE TIME SEPARATED BY:(HH:MM:SS).ASSUME THE TIME IS REPRESENTED IN 12 HOURS   (10)

IF THE INPUT VALUE EXCEEDS THE HOUR ,MINUTE AND SECOND THEN YOUR FUNCTION SHOULD REPORT AS IN VALID HOUR/MINUTE/SECOND. (20)

WRITE AN OPERATOR FUNCTION BINARY MINUS (-) TO FIND WHICH TIME IS LATER .FOR EXAMPLE INVOKING T1-T2 DISPLAYS T1 IS LATER OR T2 IS LATER (20)

```
#include<iostream>

#include<iomanip> using

namespace std;


class TIME1

{ public: int

hr,min,sec;


TIME1()

{

hr=min=sec=0;

}


TIME1(int h,int m,int s)

{ hr=h;

min=
m;
sec=s;
}

TIME1 operator -()
{
TIME1 t;
t.hr=-hr;
t.min=-min;
t.sec=-sec; return
t;
}


friend TIME1 operator -(TIME1 &t1,TIME1 &t2)
{
TIME1 t3; t3=-t2;
if((t1.hr+t3.hr)>0)
return t1; else
if((t1.hr+t3.hr)<0)
return t2; else
if((t1.hr+t3.hr)==0)   {
   if((t1.min+t3.min)>0)
return t1;    else
if((t1.min+t3.min)<0)
return t2;    else
if((t1.min+t3.min)==0)
      {
        if((t1.sec+t3.sec)>0)
return t1;    else
if((t1.sec+t3.sec)<0)
return t2;
          else
             return t1;
      }    else
cout<<"";
   } else
cout<<"";
}

void display()
{
cout<<setfill('0')<<setw(2)<<hr<<":";
cout<<setfill('0')<<setw(2)<<min<<":"; cout<<setfill('0')<<setw(2)<<sec;
}
```

```
};

void except(int hr,int min,int sec)
{

if(hr>12)
throw(hr);
if(min>59)
throw(min);
if(sec>59)
throw(sec);

}

int main()
{
        int flag=0;    int
h1,h2,m1,m2,s1,s2;
cout<<"\nenter the time1:";

         cin>>h1>>m1>>s1;

         try
         {
                  except(h1,m1,s1);
         }


         catch(int a)
         {
               cout<<"enter 12 hrs format time"<<endl;
flag=1;
         }


         if(flag==0)
         {
                     TIME1 t1(h1,m1,s1);
cout<<"enter the time2:";
               cin>>h2>>m2>>s2;
               try
                {
                     except(h2,m2,s2);
                }
                  catch(int a)
                     {
                          cout<<"enter 12hrs format time"<<endl;
flag=1;
                     }
if(flag==0)
```

```
                        {
                          TIME1 t2(h2,m2,s2);
cout<<"t1=";
t1.display();
cout<<"\nt2=";
t2.display();
cout<<"\n";                          TIME1
t3;
                         t3=t1-t2;  t3.display();  cout<<"  is later";
return 0;
                      }
           }
}
```

(XIX) Write program to calculate the area of polygons namely, Square, Rectangle, Triangle, Equilateral
Triangle, Parallelogram, Trapezoid and circle. (Use function overloading) Area of the polygons is given below.
Square A = side * side, where side is in integer variable
Rectangle = length * breadth, where length and breadth are integers
Circle = pi * radius * radius, where radius is a float variable
Triangle = 0.5 * breadth * height, where breadth and height are integer and float variables
Equilateral Triangle = 0.5 * breadth * height, where breadth and height are float and integer variables
Trapezoid = base1 * base2 * height, base1, base2 and height are integer variables
Write main() program to demonstrate the function overloading (30)

```
#include<iostream>

using namespace std;

#define pi 3.14

#include<conio.h> class

funoverload

{      public:


void area (int side)

{

        cout<<side*side;

}

void area(int length,int breadth)

{

        cout<<length*breadth;
}
void area(float radius)
{
        cout<<pi*radius*radius;
}
void area(int breadth,float height)
{
        cout<<0.5*breadth*height;
}
void area(float breadth,int height)
{
        cout<<0.5*breadth*height;
}
void area(int base1,int base2,int height)
{
        cout<<base1*base2*height;
} };  int
main()
{
      funoverload f;
      int ch;          int
s,b1,b2;     float
fb,fh;        int l,b,h;
```

```
        float r;
cout<<"Enter your
choice:1.Square
2.Rectangle 3.Circle
4.Triangle 5.Eq
triangle 6.Trapezoid 7.exit"; while(1)
{
        cout<<"Enter your choice!";
                cin>>ch;      switch(ch)
          {
                  case 1:
                          cout<<"Enter side";
                          cin>>s;
                          cout<<"Area is:"
                        f.area(s);
                        break;
                case 2:
                        cout<<"Enter length and breadth";
        cin>>l>>b;
                          cout<<"Area is:"
                          f.area(l,b);
                        break;
                case 3:
                          cout<<"Enter radius";

                          cin>>r;

                          cout<<"Area is:"

                        f.area(r);

                        break;

                case 4:

                          cout<<"Enter breadth and height";

                        cin>>b>>fh;

        cout<<"Area is:"

                          f.area(b,fh);

                        break;

                case 5:

                          cout<<"Enter breadth and height";

                        cin>>fb>>h;

        cout<<"Area is:"

                          f.area(fb,h);

                        break;

                case 6:

                          cout<<"Enter base1,base2,height";

                        cin>>b1>>b2>>h;

        cout<<"Area is:"

                          f.area(b1,b2,h);
```

```
                        break;

                case 7:

        exit(0);



                          }

}


          return(0);

}
```

// Create a text file sample.txt. Write ten sentences into sample.txt, read and display them by opening it in read mode.

```
#include<iostream>

#include<fstream>

using namespace std;

int main() { ofstream

fout;

fout.open("sample.txt")

;

char str[500]="This is CSE-B";      //writing into file

fout<<str; fout.close(); ifstream fin;

fin.open("sample.txt"); char ch;
while(!fin.eof())
{
   fin.get(ch);  //read
if(fin.eof())    break;
   else
cout<<ch;       //display
}
fin.close();



return 0;
}
```

(XX) 1. Create a class person with the data members *name, address of type char acter array. Create a class employee derived from class person with the data mem bers basic pay, allowances, pf, gross and netpay of type integer. Create a class s tudent derived from class person with the data members course, grade. Creat e a class parttime_student derived from student and employee with the data me mbers stipend and total_income of type integer. •Model this relationship usin g inheritance. Write a program to demonst rate the type of inheritance (20)

• Allocate memory for the data member 'name' using constructor (10)

• Deallocate the memory for 'name' using destructor (10)

• Write suitable exceptions for the given scenario (10)

```
#include <iostream>

#include<stdlib.h>

using namespace std;

class person  {

public:

            char* name;

int length;                char

address[20];

person()                   {

                      length=0;

                       name=new char[length+1];

              }

void getper()

             {

                       cout<<"\n Enter name ";

cin>>name;

                       cout<<"\n Enter address ";

cin>>address;

              }

              ~person()

               {

                          delete name;

                }
```

```
 };
 class employee:virtual public person
{
public:
            int bp,a,pf,g,np;
void gete()
            {
                          cout<<"\n Enter basic pay,allowances,pf,gross,netpay ";
cin>>bp>>a>>pf>>g>>np;                                    try
{
                                    if(bp==0)
                                   {
                                             throw(bp);
                                    }
}
                           catch(int b)
                           {
                                      cout<<"\n Error bp=0 ! Try with different bp ";
exit(0);

                            }
                 }
};
 class student:virtual public person
```

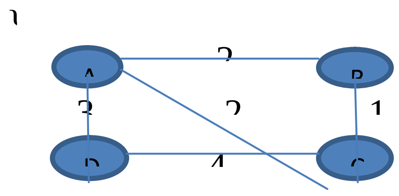
```
{
public:
            char course[10];
int grade;             void
getst()
            {
                      cout<<"\n Enter course,grade :";
cin>>course>>grade;
             }
 };
 class parttime_student:public student,public employee
{
public:
                int st,ti;
void getpa()
                {
                             cout<<"\n Enter stipend ";
cin>>st;
                 }
                 void display()
                 {
                            ti=st+bp+a+pf+g+np;
                            cout<<"\tName\tAddress\tCourse\tGrade\tTotal income\n";
```

```
                              cout<<"\t"<<name<<"\t"<<address<<"\t";
  cout<<"\t"<<grade<<"\t"<<ti<<"\n";
                        }
   };  int
  main()
   {
        parttime_student part;
  part.getper();
  part.gete();
  part.getst();
  part.getpa();
  part.display();       return
  0;
   }
```

1. **Prims Algorithm:**

```
#include<iostream.h> #include<conio.h> int a,b,u,v,n,i,j,ne=1;
int visited[10]={0},min,mincost=0,cost[10][10]; class
graph
{
public:
void prim()
{
```

```
cout<<"\n"<<" Enter the number of vertices:";  cin>>n;

 cout<<"\n"<<" Enter the adjacency matrix:"<<"\n";  for(i=1;i<=n;i++)

  for(j=1;j<=n;j++)

  {

   cin>>cost[i][j];    if(cost[i][j]==0)

    cost[i][j]=999;

  }

 visited[1]=1;  cout<<"\n";

 while(ne<n)

 {

  for(i=1,min=999;i<=n;i++)
for(j=1;j<=n;j++)     if(cost[i][j]<min)
if(visited[i]!=0)

     {

      min=cost[i][j];       a=u=i;

      b=v=j;

     }

  if(visited[u]==0 || visited[v]==0)

  {

   cout<<"\n"<< "Edge"<<ne++<<"\t"<<a<<"-->"<<b<<"\t"<<"cost:"<<min;    mincost+=min;

   visited[b]=1;

  }

  cost[a][b]=cost[b][a]=999;

 }
 cout<<"\n";  cout<<"Mincost is
:"<<mincost;

 }  };

int main()

{  clrscr();  graph
g;

 g.prim();  getch();  return
0;
```



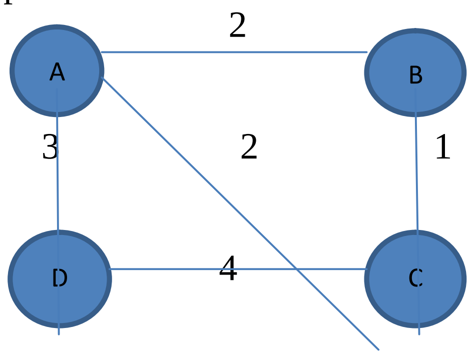
2. <u>Kruskal Algorithm</u>

```
#include<iostream.h>

#include<conio.h>
#include<stdlib.h> int
i,j,k,a,b,u,v,n,ne=1;

int min,mincost=0,cost[9][9],parent[9]; class graph

{
```

```
public:

void kruskal()

{

cout<<"\n"<<"Implementation of Kruskal's algorithm"<<"\n";

 cout<<"\n"<<"Enter the no. of vertices\n";  cin>>n;

 cout<<"\n"<<"Enter the cost adjacency matrix"<<"\n";  for(i=1;i<=n;i++)

 {

  for(j=1;j<=n;j++)

  {

   cin>>cost[i][j];    if(cost[i][j]==0)
cost[i][j]=999;

  }

 }

 cout<<"\n"<<"The edges of Minimum Cost Spanning Tree are"<<"\n";  while(ne<n)

 {

  for(i=1,min=999;i<=n;i++)

  {

   for(j=1;j<=n;j++)

   {

    if(cost[i][j]<min)

    {

     min=cost[i][j];      a=u=i;

     b=v=j;

    }
   }

  }

  u=find(u);   v=find(v);
if(uni(u,v))

  {

   cout<<"\n"<<ne++<<"\t"<< "edge"<<a<<"--->"<<b<<"="<<min;    mincost +=min;

  }

  cost[a][b]=cost[b][a]=999;

 }

 cout<<"\n"<<"Minimum cost ="<<mincost          ;

 }

 int find(int i)

{

 while(parent[i])   i=parent[i];

 return i;

}

int uni(int i,int j)

{  if(i!=j)

 {

  parent[j]=i;

  return 1;

 }

 return 0;

} };
```

```
void main()

{  clrscr();  graph
g;

 g.kruskal();

 getch();

}
```

Output:
Graph:

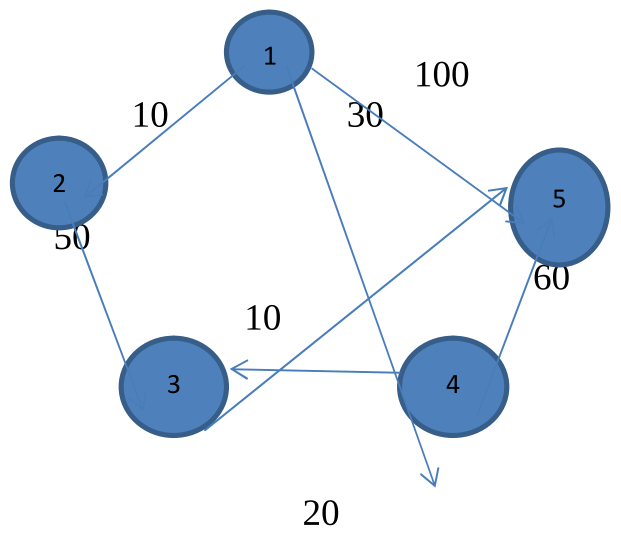


3. <u>Dijkstra's Algorithm</u>

```
#include<iostream.h>

#include<conio.h>       #define
infinity 999 class dijkstra
{

public: void dij(int n,int v,int cost[10][10],int dist[])

{

 int i,u,count,w,flag[10],min;
for(i=1;i<=n;i++)
flag[i]=0,dist[i]=cost[v][i];  count=2;
while(count<=n)

 {
  min=99;   for(w=1;w<=n;w++)
if(dist[w]<min && !flag[w])
min=dist[w],u=w;   flag[u]=1;
count++;   for(w=1;w<=n;w++)

   if((dist[u]+cost[u][w]<dist[w]) && !flag[w])
dist[w]=dist[u]+cost[u][w];

 }

} };

void main()

{

 int  n,v,i,j,cost[10][10],dist[10];    clrscr();
dijkstra d;
 cout<<"\n"<<"Enter the number of vertices:";  cin>>n;

 cout<<"\n"<<"Enter the cost matrix:"<<"\n";
for(i=1;i<=n;i++)   for(j=1;j<=n;j++)

  {

   cin>>cost[i][j];    if(cost[i][j]==0)
cost[i][j]=infinity;

  }

 cout<<"\n"<<"Enter the source matrix:";  cin>>v;

 d.dij(n,v,cost,dist);  cout<<"\n"<<"Shortest
path:"<<"\n";  for(i=1;i<=n;i++)   if(i!=v)

   cout<<v<<"-->"<<i<<"cost"<<dist[i]<<"\t";  getch();

}
```

Output:

Graph:



4. **Breadth First Traversal**

```
#include<iostream.h>
#include<conio.h> int
visited[10],visited1[10]; class que

{

        protected:

                int q[20],front,rear;
public:                 void enqueue(int x)

                {

                                rear=rear+1;
q[rear]=x;                              if(front==-1)

                                {

                                        front=front+1;

                                }

                }

                int dequeue()
```

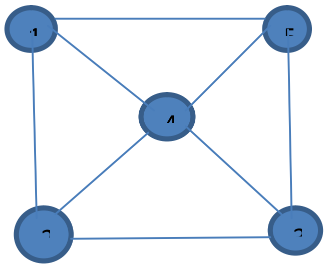
```
                    {                              int x;
                              x=q[front];

                                        front=front+1;
return(x);

                   }

                    int isempty()

                   {

                             if(front>rear)

                                        return(1);
else

                                       return(0);

                   }

};

class graph:public que

{

          int n,g[10][10];      public:

                   graph()

                   {

                             int i;

                              cout<<"Enter the number of vertices:"<<endl;

                             cin>>n;

                              for(i=1;i<=n;i++)
visited[i]=0;                               for(i=0;i<=n;i++)

                                        q[i]=0;
front=-1;                               rear=-1;
```

```
                }

                void getgraph()

                {

                        int i,j;

                         cout<<"Enter the adjacency  matrix:"<<endl;

                         for(i=1;i<=n;i++)
for(j=1;j<=n;j++)                                   cin>>g[i][j];

                        bfs(1);

                }


                void bfs(int u)

                {

                        visited[u]=1;

                        cout<<"BFS:";

                        enqueue(u);


                        while(!isempty())

                        {

                                int v;

                                 u=dequeue();
//cout<<"BFS:";                             cout<<u<<" ";
                 for(v=1;v<=n;v++)

                                {

                                        if(g[u][v]==1)

                                        {
```

```
                                        if(visited[v]==0)

                                        {

                                              enqueue(v);
                         visited[v]=1;

                                        }

                                   }

                              }

                         }

                    }


};

void main()

{

     clrscr();      graph obj;

     cout<<"Breadth first output:"<<endl; obj.getgraph();


     getch();

}
```

Output:

Graph:



5. Depth First Traversal

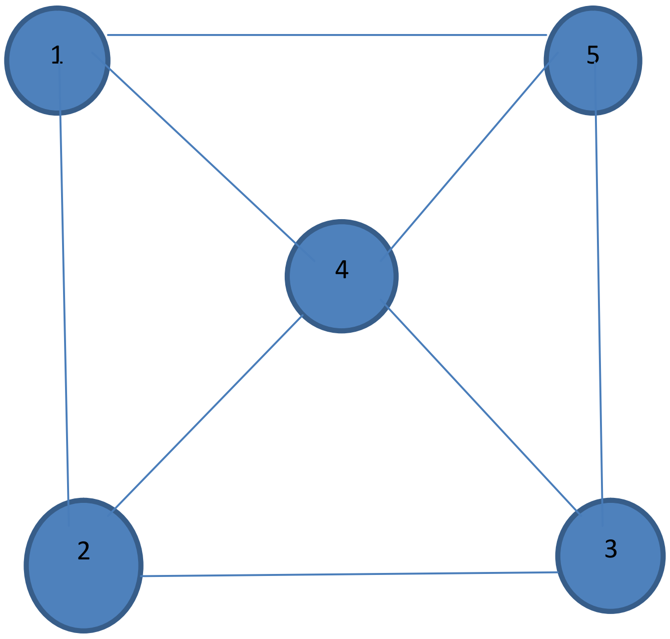
```
#include<iostream.h>

#include<conio.h>

int visited[10],visited1[10]; class que

{

          protected:

                   int q[20],front,rear;
public:                      void enqueue(int x)

                  {

                                   rear=rear+1;
q[rear]=x;                                 if(front==-1)

                                  {

                                           front=front+1;
```

```
                    }
            }
            int dequeue()
            {
                    int x;
                     x=q[front];
front=front+1;
                    return(x);
            }
            int isempty()
            {
                if(front>rear)
                     return(1);
else
                    return(0);
            }
};
class graph:public que
{
        int n,g[10][10];    public:
            graph()
            {
                int i;
```

```
                              cout<<"Enter the number of vertices:"<<endl;
cin>>n;

                              for(i=1;i<=n;i++)
visited[i]=0;                                 for(i=0;i<=n;i++)

                                        q[i]=0;
front=-1;                               rear=-1;

                    }

                    void getgraph()

                    {

                              int i,j;

                              cout<<"Enter the adjacency  matrix:"<<endl;

                              for(i=1;i<=n;i++)
for(j=1;j<=n;j++)

                                                cin>>g[i][j];


                    }



                    void initial()

                    {

                              int i;

                             for(i=1;i<=n;i++)

                                       visited1[i]=0;

                    }

                    void dfs(int v)

                    {
                        int u,i;
visited1[v]=1;                      cout<<v<<" ";
          for(u=1;u<=n;u++)

                        {

                              if(g[v][u]==1)

                              {

                                   if(visited1[u]==0)

                                        dfs(u);

                              }

                        }

                   }


};


void main()

{

     clrscr();        graph obj;


obj.getgraph();       cout<<"Depth first
output:"<<endl;       obj.initial();        obj.dfs(1);
getch();

}
```

Output:

Graph:



IMPLEMENTATION OF BINARY SEARCH TREE

//binary search tree

```
#include<iostream.h>

#include<conio.h>

#include<alloc.h>
#include<stdlib.h> struct treenode

{

int element; struct treenode *left;
struct treenode *right;

}*right=NULL,*left=NULL;


typedef struct treenode *tree;



class binsearchtree

{
```

```
public:
            tree insert(int x,tree T)
           {
            struct treenode *temp;
           temp=(struct treenode *)malloc(sizeof(struct treenode));
           if(temp==NULL)
           {
           cout<<"out of space";
           }
           else
           {
           if(T==NULL)
           {
           temp->element=x;            temp->left=NULL;
temp->right=NULL;
           T=temp;
           }
           else
           {
                       if(x<T->element)                        T-
>left=insert(x,T->left);
                      else
                                  T->right=insert(x,T->right);
           }
```

```
}

return T;

}




tree Delete(int x,tree T)

{

          tree tmpcell;          if(T==NULL)

                    cout<<"No elements in the tree"<<endl;

          else

          {

                    if(x<T->element)                      T-
>left=Delete(x,T->left);             else if(x>T->element)
          T->right=Delete(x,T->right);

                    else if(T->left && T->right)

                    {

                              tmpcell=Findmin(T->right);

                              T->element=tmpcell->element;                              T->right=Delete(T-
          >element,T->right);

                    }

                    else

                    {

                              tmpcell=T;                      if(T-
>left==NULL)                                   T=T->right;

                              else if(T->right==NULL)
```

```
                                        T=T->left;

                              free(tmpcell);

                    }

          }

           return(T);

}


void display(tree T)        //inorder

{

  if(T!=NULL)

  {

  display(T->left);   cout<<T->element<<" ";
display(T->right);

  }

 }


tree Find(int x,tree T)

{

  if(T==NULL)

          return(T);

  if(x<T->element)           return(Find(x,T-
>left));   else if(x>T->element)
return(Find(x,T->right));

  else

          return(T);
```

```
}

tree Findmax(tree T)
{
  if(T==NULL) return(NULL);
  else if(T->right==NULL)        return(T);
  else
           return(Findmax(T->right));
}

tree Findmin(tree T)
{
  if(T==NULL)
           return(NULL);
  else if(T->left==NULL)
           return(T);
  else
           return(Findmin(T->left));
}

void inorder(tree T)
{
 if(T!=NULL)
 {
```

```
 inorder(T->left);  cout<<T->element<<" ";

 inorder(T->right);

 }

}


void preorder(tree T)

{

 if(T!=NULL)

 {

 cout<<T->element<<" ";  preorder(T->left);
preorder(T->right);

 }

}


void postorder(tree T)

{

 if(T!=NULL)

 {

postorder(T->left);  postorder(T->right);

 cout<<T->element<<" ";

 }

}


};
```

```
main()

{

  binsearchtree BST;   int
n,x,i,choice,a;

  tree T=NULL,temp;   cout<<"Enter the no of
nodes:";   cin>>n;

  for(i=0;i<n;i++)

  {

  cin>>x;

  T=BST.insert(x,T);

  }

  BST.display(T);

  do

  {

  cout<<endl<<"The options available are:"<<endl;

  cout<<"1.Inorder  2. Preorder 3.Postorder  4.Find 5. Findmax 6. Findmin

7.Delete  8.Exit \n ";

  cout<<"Enter the choice  of  traversal to display";
cin>>choice;   switch(choice)

  {   case 1:

          BST.inorder(T);

          break;   case 2:

          BST.preorder(T);

          break;   case 3:
```

```
          BST.postorder(T);

          break;   case 4:

         cout<<"Enter the element to be searched:"<<endl; cin>>a;

          temp=BST.Find(a,T);      if(temp!=NULL)
cout<<"Element found!!"<<endl;

          else

          cout<<"Element not found!!"<<endl;

          break;   case 5:

          temp=BST.Findmax(T);  cout<<"max element is
"<<temp->element;          break;   case 6:

          temp=BST.Findmin(T);   cout<<"min element is
"<<temp->element;

          break;   case 7:

           cout<<"Enter the element to be deleted:"<<endl;

          cin>>a;

          T=BST.Delete(a,T);         BST.display(T);

          break;  case 8:

          exit(0);   default:

         cout<<"invalid choice";

  }

  }while(1);


}
```