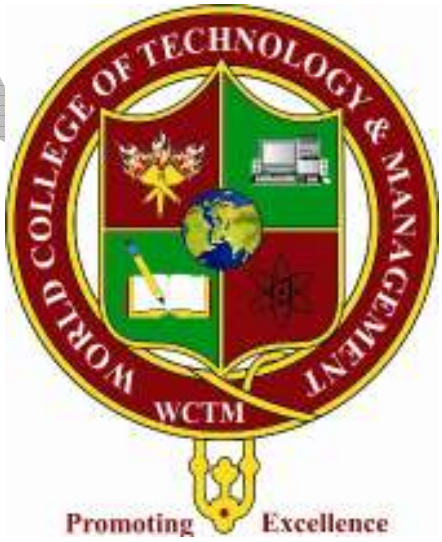
LAB MANUAL
FOR
OOPS LAB



Raising a number n to a power p is same as multiplying n by itself p times. Write a function called power that takes a double value for n and an int value for p returns the result as double value. Use a default argument of 2 for p so that if this argument is omitted the number will be squared. Write the main function that gets the values from user the user to test the function.

ALGORITHM

1. Start.
2. Declare a function double power with two parameters one of type double and other of type integer.
3. Start the main function.
4. Declare two variables n, res of type double one variable p of type integer and one variable ch of type character.
5. Clear the screen.
6. Ask user the number, which he wants to raise to a power.
7. Get that value in n.
8. Decide weather the user wants to enter the power.
9. Get the input in char type ch.
10. If ch =’y’ or ’Y’ than go to step number 12.
11. Else go to step number 17.
12. If ch = ‘y’ or ‘Y’.
13. Enter the power number.
14. Get the input in p.
15. Call the function double power and pass two parameters.
16. End of step 12 if structure.
17. If ch is any other value than.
18. Call the function double power and pass only one parameter that is the number that has to be raised.
19. End of step 17 if structure.
20. Print the result.
21. Stop.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
double power(double n,int p=2)
{
        double x=1.0;
        int i;
        for (i=0;i<p;i++)
                x*=n;
        return x;
}
void main()
{
        double n,res;
        int p;
        char ch;
        clrscr();
        cout<<"Enter a number to be raised to a power: ";
        cin>>n;
        cout<<"Do you want to enter the power (Y/N): ";
        cin>>ch;
        if ((ch=='y')||(ch=='Y'))
        {
                cout<<"Enter the power to which the number "<<n<<" is to be raised: ";
                cin>>p;
                res=power(n,p);
        }
        else
                res=power(n);
        cout<<"The result is: "<<res;
        getch();
}
```

OUTPUT

Enter a number to be raised to a power: 5

Do you want to enter the power (Y/N): Y

Enter the power to which the number 5 is to be raised: 3

The result is: 125

A point on the 2-d can be represented by two numbers: an x co-ordinate and a y co-ordinate. For example (4,5) represents a point 4 units to the right of origin and 5 units up the origin. The sum of two points can be defined as a new point whose x co-ordinate is the sum of x co-ordinates of both points and same for y co-ordinates. Write a program that uses a structure called point to model a point. Define three points, and have the user input values to two of them. Then set the third point equal to the sum of the other two and display the value of the new point.

ALGORITHM

1. Start.
2. Declare a structure named point with two variables x, y of type integer.
3. Start the main function.
4. Declare three variables p1, p2, p3 of type point (of the name of structure).
5. Clear the screen.
6. Get the input of first point from user in variable p1 of type point, the input would be in p1.x and p1.y.
7. Get the input of second point from user in variable p2 of type point, the input would be in p2.x and p2.y.
8. P3.x = p1.x + p2.x.
9. P3.y = p1.y + p2.y.
10. Display p3.x and p3.y.
11. Stop.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
struct point
{
        int x,y;
};
void main()
{
        point p1,p2,p3;
        clrscr();
        cout<<"Enter coordinates for P1: ";
        cin>>p1.x>>p1.y;
        cout<<"Enter coordinates for P2: ";
        cin>>p2.x>>p2.y;
        p3.x=p1.x+p2.x;
        p3.y=p1.y+p2.y;
        cout<<"Coordinates of P1 + P2 are: "<<p3.x<<" "<<p3.y;
        getch();
}
```

OUTPUT

Enter coordinates for P1: 2 3

Enter coordinates for P2: 4 6

Coordinates of P1 + P2 are: 6 9

Create the equivalent four-function calculator. The program should request the user to enter a number, an operator, and another number. It should then carry out the specified arithmetical operation: adding, multiplying, or dividing the two numbers. (It should use a switch statement to select the operation). Finally it should display the result.

When it finishes the calculation, the program should ask if the user wants to do another calculation. The response can be 'Y' or 'N'.

ALGORITHM

1. Start.
2. Declare a class named calc with following variables and member functions in private region.
   I. Two variables x and y of float type.
   II. One variable op of char type.
   III. One variable res of double type.
   IV. Member function void sum ( ).
   V. Member function void sub ( ).
   VI. Member function void mul ( ).
   VII. Member function void div ( ).
   VIII. Member function void disp ( ).
3. Declare the following member functions in public part.
   I. Member function void input ( ).
   II. Member function void eval ( ).
4. Close the class.
5. Start the main function.
6. Declare c as an object of class calc.
7. Declare a variable ch of char type.
8. Repeat steps 9 to 12 till ch = 'y' or ch = 'y'.
9. Clear the screen.
10. Call the input member function of class calc using object c of class calc.
11. Call the evaluate member function of class calc using object c of class calc.
12. Ask user for another operation.
13. End of step 8 loop.
14. Stop.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
class calc
{
        float x,y;
        char op;
        double res;
        void sum();
        void sub();
        void mul();
        void div();
        void disp();
        public:
                void input();
                void eval();
};
void calc::input()
{
        cout<<"Enter first number, operator, second number: ";
        cin>>x>>op>>y;
}
void calc::eval()
{
        switch (op)
        {
                case '+':       sum();
                                        disp();
                                        break;
                case '-' :      sub();
                                        disp();
                                        break;
                case '*':       mul();
                                        disp();
                                        break;
                case '/' :      div();
                                        disp();
                                        break;
                default :       cout<<"Invalid operator";
        }
}
```

```
void calc::sum()
{
        res=x+y;
}
void calc::sub()
{
        res=x-y;
}
void calc::mul()
{
        res=x*y;
}
void calc::div()
{
        res=x/y;
}
void calc::disp()
{
        cout<<"Answer = "<<res;
}
void main()
{
        calc c;
        char ch;
        do
        {
                clrscr();
                c.input();
                c.eval();
                cout<<"\nDo another (Y/N)? ";
                cin>>ch;
        }while ((ch=='y')||(ch=='Y'));
}
```

OUTPUT

Enter first number, operator, second number: 4+9

Answer = 13

Do another (Y/N)? Y

Enter first number, operator, second number: 5*9

Answer = 45

Do another (Y/N)? N

A phone number , such as (212) 767-8900, can be thought of as having three parts: the area code (212), the exchange (767) and the number (8900). Write a program that uses a class to store these three parts of a phone number separately. Call the class phone. Create two class objects of type phone. Initialize one, and have the user input a number for the other one. Display both the numbers.

ALGORITHM

1. Start.
2. Declare a class named phone with following variables in private region.
   I. Two variables area, exc of int type.
   II. One variable num of type long.
3. Declare the following in public part.
   I. A constructor with three parameters phone (int, int, long).
   II. A default contructor.
   III. Member function void input ( ).
   IV. Member function void disp ( ).
4. Close the class.
5. Start the main function.
6. Declare two objects p1(91, 124, 2304871), p2 of class they will automatically invoke constructors.
7. Clear the screen.
8. Call input member function of class phone through object of class p2.
9. Print "my number is = ".
10. Call disp member function of class phone through object of class p1.
11. Print "your number is = ".
12. Call disp member function of class phone through object of class p2.
13. Get the result.
14. Stop.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
class phone
{
        int area,exc;
        long num;
        public:
                phone(int,int,long);
                phone()
                {
                }
                void input();
                void disp();
};
phone::phone(int a,int b,long c)
{
        area=a;
        exc=b;
        num=c;
}
void phone::input()
{
        cout<<"Enter our area code, exchange, and number: ";
        cin>>area>>exc>>num;
}
void phone::disp()
{
        cout<<"("<<area<<")"<<exc<<"-"<<num;
}
void main()
{
        phone p1(91,124,2304871),p2;
        clrscr();
        p2.input();
        cout<<"My number is ";
        p1.disp();
        cout<<"\nYour number is ";
        p2.disp();
        getch();
}
```

OUTPUT

Enter our area code, exchange, and number: 91 124 2322292

My number is (91)124-2304871

Your number is (91)124-2322292

Create two classes DM and DB which store the value of distances. DM stores distance in meters and centimeters and DB in feet and inches. Write a program that can read values for the class objects and add one object of DM with another object of DB.
Use a friend function to carry out the addition operation. The object that stores the results may be a DM object or DB object, depending on the units in which the results are required.
The display should be in the format of feet and inches or meters and centimeters depending on the object on display.

ALGORITHM

1. Start
2. Declare a class DM with following variables in private region.
   I. Two variables met, cent of type int.
3. Declare the following in public part.
   I. Member function void input ( ).
   II. Member function void disp ( ).
   III. Friend member function friend void sum (DM&, DB).
   IV. Friend member function friend void sum (DB&, DM).
4. Close the class.
5. Declare another class DB with following variables in private region.
   I. Two variables met, cent of type int.
6. Declare the following in public part.
   V. Member function void input ( ).
   VI. Member function void disp ( ).
   VII. Friend member function friend void sum (DM&, DB).
   VIII. Friend member function friend void sum (DB&, DM).
7. Close the class.
8. Start the main function.
9. Declare d1 an object of class DM.
10. Declare d2 an object of class DB.
11. Declare a variable a of int type.
12. Clear the screen.
13. Call input member function of class DM through object d1.
14. Call input member function of class DB through object d2.
15. Print 1 to get output in meters and 2 to get output in feet.
16. Get the output in a.
17. If a=1 go to step number 18 else go to step number 22.
18. If a = 1.
19. Call function sum (d1, d2).
20. Call disp member function of class DM through the object d1 of class DM.
21. End of step 18 if structure.
22. If a = 2.

23. Call function sum (d2, d1).
24. Call disp member function of class DB through the object d2 of class DB.
25. End of step 22 if structure.
26. Get the result.
27. Stop.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
class DB;
class DM
{
        int met,cent;
        public:
        void input();
        void disp();
        friend void sum(DM&,DB);
        friend void sum(DB&,DM);
};
void DM::input()
{
        cout<<"Enter distace in metres, centimeters: ";
        cin>>met>>cent;
}
void DM::disp()
{
        cout<<"The distance in metres and centimeters is: "<<met<<"metres and
"<<cent<<"centimeters";
}
class DB
{
        int feet,inch;
        public:
        void input();
        void disp();
        friend void sum(DM&,DB);
        friend void sum(DB&,DM);
};
void DB::input()
{
        cout<<"Enter distace in feet, inches: ";
        cin>>feet>>inch;
}
void DB::disp()
{
        cout<<"The distance in feet and inches is: "<<feet<<"feet and
"<<inch<<"inches";
}
void sum(DM& dm,DB db)
```

```
{
        float x=db.feet*30.48;
        x+=db.inch*2.54;
        int y=x;
        dm.met+=y/100;
        dm.cent+=y%100;
        dm.met+=dm.cent/100;
        dm.cent=dm.cent%100;
}
void sum(DB& db,DM dm)
{
        float x=dm.met*39.37;
        x+=dm.cent*0.3937;
        int y=x;
        db.feet+=y/12;
        db.inch+=y%12;
        db.feet+=db.inch/12;
        db.inch=db.inch%12;
}
void main()
{
        DM d1;
        DB d2;
        int a;
        clrscr();
        d1.input();
        d2.input();
        cout<<"Enter 1 to get output in metres format or enter 2 to get output in feet
format: ";
        cin>>a;
        if (a==1)
        {
                sum(d1,d2);
                d1.disp();
        }
        else
        {
                sum(d2,d1);
                d2.disp();
        }
        getch();
}
```

OUTPUT

Enter distace in metres, centimeters: 5 30

Enter distace in feet, inches: 3 6

Enter 1 to get output in metres format or enter 2 to get output in feet format: 1

The distance in metres and centimeters is: 6metres and 36centimeters

Enter distace in metres, centimeters: 5 30

Enter distace in feet, inches: 3 6

Enter 1 to get output in metres format or enter 2 to get output in feet format: 2

The distance in feet and inches is: 20feet and 10inches

Create a class rational which represents a numerical value by two double values-NUMERATOR and DENOMINATOR. Include the following public member functions:

- **Constructor with no arguments (default).**
- **Constructor with two parameters.**
- **Overload + operator to enable addition of two rational numbers.**
- **Reduce() function to reduce the rational number by eliminating the highest common factor between the numerator and the denominator.**
- **Overload >> operator to enable input through in.**
- **Overload << operator to enable output through out.**

Write a main ( ) to test all the functions in the class.

ALGORITHM

1. Start
2. Declare a class rational phone with following variables in private region.
   I. Two variables num, denum of type double.
3. Declare the following in public part.
   I. A default constructor rational ( ).
   II. A parameterized constructor rational (double, double).
   III. A member function to overload + operator to add two rational numbers rational operator+(rational).
   IV. A member function void reduce ( ).
   V. Friend function for overloading operator >> friend void operator >> (istream&, rational&).
   VI. Friend function for overloading operator << friend void operator << (ostream&, rational&).
4. Close the class.
5. Start the main function.
6. Clear the screen.
7. Declare r1, r2(10,20), r3 as objects of class rational.
8. Print r2 on screen.
9. Call member function reduce of class rational through the object r2.
10. Again print the result on screen.
11. Get the input of r1 from the user.
12. Add r1, r2 and assign the result to r3.
13. Print r3 on screen.
14. Call member function reduce of class rational through the object r3.
15. Print the result on screen.
16. Stop.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
class rational
{
        double num,denum;
        public:
        rational();
        rational(double,double);
        rational operator+(rational);
        void reduce();
        friend void operator>>(istream&,rational&);
        friend void operator<<(ostream&,rational&);
};
rational::rational()
{
        num=1;
        denum=1;
}
rational::rational(double a,double b)
{
        num=a;
        denum=b;
}
void rational::reduce()
{
        long x,y;
        x=num;
        y=denum;
        int z;
        if (num>denum)
        {
                do
                {
                        z=x%y;
                        x=y;
                        y=z;
                }while (z!=0);
                num/=x;
                denum/=x;
        }
        else
        {
```

```
                do
                {
                        z=y%x;
                        y=x;
                        x=z;
                }while (z!=0);
                num/=y;
                denum/=y;
        }
}
rational rational::operator+(rational r)
{
        rational r1;
        r1.denum=denum*r.denum;
        r1.num=(num*r.denum)+(r.num*denum);
        return r1;
}
void operator>>(istream& in,rational& r)
{
        cout<<"\nEnter numerator and denominator for the rational number: ";
        in>>r.num>>r.denum;
}
void operator<<(ostream& out,rational& r)
{
        out<<"\nThe rational number is: "<<r.num<<"/"<<r.denum;
}
void main()
{
        clrscr();
        rational r1,r2(10,20),r3;
        cout<<r2;
        r2.reduce();
        cout<<r2;
        cin>>r1;
        r3=r1+r2;
        cout<<r3;
        r3.reduce();
        cout<<r3;
        getch();
}
```

OUTPUT

The rational number is: 10/20

The rational number is: 1/2

Enter numerator and denominator for the rational number: 4 2

The rational number is: 10/4

The rational number is: 5/2

Write a program that creates a binary file by reading the data for the students from the terminal. The data of each student consist of roll no., name (a string of 30 or lesser no. of characters) and marks.

ALGORITHM

1. Start
2. Declare a class stu with following in private region.
   I. Two variables rno, marks of type int
   II. One array named nm of 31 characters of type char.
   III. A member function void write ( ).
3. Declare the following in public part.
   I. A member function void input ( ).
   II. A member function void display ( ).
4. Close the class.
5. Start the main function.
6. Declare s1 as an object of class stu.
7. Declare a variable ch of char type.
8. Open a file sta.dat in binary mode using object of class ofstream as fout.
9. Close the binary file sta.dat using object fout.
10. Call the member function input of class stu using its object s1.
11. Input ch.
12. Repeat steps 10 and 11, if ch=y.
13. Call the member function display of class stu using its object s2.
14. Stop.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
#include<stdio.h>
#include<string.h>
class stu
{
        int rno,marks;
        char nm[31];
        void write();
        public:
                void input();
                void display();
};
void stu::write()
{
        stu s;
        s.rno=rno;
        s.marks=marks;
        strcpy(s.nm,nm);
        ofstream fout("sta.dat",ios::app|ios::binary);
        fout.write((char*)&s,sizeof(s));
        fout.close();
}
void stu::input()
{
        clrscr();
        cout<<"Enter student's roll number: ";
        cin>>rno;
        cout<<"Enter student's name: ";
        gets(nm);
        cout<<"Enter student's marks: ";
        cin>>marks;
        write();
}
void stu::display()
{
        clrscr();
        int y;
        stu s;
        cout<<"The information of the students stored in the binary file is:\n\n";
        ifstream fin("sta.dat");
```

```
        fin.read((char*)&s,sizeof(s));
        while(!fin.eof())
        {
                cout<<s.rno;
                y=wherey();
                gotoxy(7,y);
                cout<<s.nm;
                gotoxy(40,y);
                cout<<s.marks<<endl;
                fin.read((char*)&s,sizeof(s));
        }
        fin.close();
}
void main()
{
        stu s1;
        char ch;
        ofstream fout("sta.dat",ios::binary);
        fout.close();
        do
        {
                s1.input();
                cout<<"\n\nDo you want to enter more (Y/N): ";
                cin>>ch;
        }while ((ch=='y')||(ch=='Y'));
        s1.display();
        getch();
                                        }
```

OUTPUT

Enter student's roll number: 6019
Enter student's name: jagmeet singh
Enter student's marks: 90

Do you want to enter more (Y/N): y

Enter student's roll number: 6007
Enter student's name: arjun grover
Enter student's marks: 85

Do you want to enter more (Y/N): y

Enter student's roll number: 6066
Enter student's name: ujjwal walia
Enter student's marks: 80

Do you want to enter more (Y/N): n

The information of the students stored in the binary file is:

6019	jagmeet singh	90
6007	arjun grover	85
6066	ujjwal walia	80

Make a class Employee with a name and salary. Make a class Manager inherit from Employee. Add an instance variable, named department, of type string. Supply a method to string that prints the manager's name, department and salary. Make a class Executive inherit from Manager. Supply a method to string that prints the string "Executive" followed by the information stored in the Manager super class object. Suppl a test program that tests these classes and methods.

ALGORITHM

1. Start
2. Declare a class employee with following in protected part.
   I. Declare an array nm of 30 characters.
   II. Declare a variable sal of float tyoe.
   III. There is no private or public part in this class.
3. Close the class.
4. Declare a second class manager where class employee is publicly inherited and it following details of its protected part.
   I. Declare an array dep of 10 characters.
5. Class manager has following declerations in public part.
   I. A member function void input ( ).
   II. A member function void disp ( ).
6. Close the class manager.
7. Declare a third class executive which has class employee inherited publicly and it has following details in its public part.
   I. A member function void disp (manager) with manager as parameter.
8. Close the class executive.
9. Start the main function.
10. Clear the screen.
11. Declare m1 as object of class manager, e as object of class executive.
12. Call input member function of class manager using m1 ,the object of class manager.
13. Clear the screen.
14. Call disp member function of class manager using m1, object of class manager.
15. C
16. Get the result.
17. Stop.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
class employee
{
        protected:
                char nm[30];
                float sal;
};
class manager:public employee
{
        protected:
                char dep[10];
        public:
                void input();
                void disp();
};
void manager::input()
{
        cout<<"Enter name: ";
        gets(nm);
        cout<<"Enter salary: ";
        cin>>sal;
        cout<<"Enter department: ";
        gets(dep);
}
void manager::disp()
{
        cout<<"\nManager's name: "<<nm;
        cout<<"\nSalary: "<<sal;
        cout<<"\nDepartment: "<<dep<<endl;
}
class executive:public manager
{
        public:
                void disp(manager);
};
void executive::disp(manager m)
{
        cout<<"\n\nExecutive";
        m.disp();
}
```

```
void main()
{
        clrscr();
        manager m1;
        m1.input();
        executive e;
        clrscr();
        m1.disp();
        e.disp(m1);
        getch();
}
```

OUTPUT

Enter name: jagmeet singh

Enter salary: 99999

Enter department: accounts

Manager's name: jagmeet singh

Salary: 99999

Department: accounts

Executive

Manager's name: jagmeet singh

Salary: 99999

Department: accounts

Imagine a tollbooth with a class called tollbooth. The two data items are: a toe unsigned int to hold the total number of cars, and a type double to hold the total amount of money collected. A constructor initializes both these to 0. A member function called paying car ( ) increments the car total and adds 0.50 to the cash total. Another function, called nonpay car ( ), increments the car total but adds nothing to the cash total. Finally a member function called displays the two totals.

Include a program to test this class. This program should allow the user to push one ke to count a paying car and another to count a nonpaying car. Pushing the ESC key should cause the program to print out the total cars and the total cash and then exit.

ALGORITHM

1. Start.
2. Declare a class tollbooth with following in private part.
   I. A variable tnc of unsigned int type.
   II. A variable tot of double type.
3. Declare the following in public part.
   I. A default constructor tollbooth ( ).
   II. A member function void payingcar ( ).
   III. A member function void nonpaycar ( ).
   IV. A member function void display ( ).
4. Close the class.
5. Start the main function.
6. Declare a variable ch of char type.
7. Declare t as an object of class tollbooth.
8. Start of do – while loop.
9. Clear the screen.
10. Get the input from the user in ch.
11. If ch = p go to step number 12or if ch = n go to step number 13.
12. Call the member function payingcar through the object t of class tollbooth.
13. Call the member function nonpaycar through the object t of class tollbooth.
14. Do this while ch != 27
15. End of do – while loop.
16. Clear the screen.
17. Call the member function display through the object t of class tollbooth.
18. Get the result.
19. Stop.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
class tollbooth
{
        unsigned int tnc;
        double tot;
        public:
                tollbooth()
                {
                        tnc=0;
                        tot=0;
                }
                void payingcar();
                void nopaycar();
                void display();
};
void tollbooth::payingcar()
{
        tnc++;
        tot+=0.50;
}
void tollbooth::nopaycar()
{
        tnc++;
}
void tollbooth::display()
{
        cout<<"\nTotal number of cars: "<<tnc;
        cout<<"\nTotal ammount collected: Rs."<<tot;
}
void main()
{
        char ch;
        tollbooth t;
        do
        {
                clrscr();
                cout<<"\nPress p for paying car"
                        <<"\nPress n for nopay car"
                        <<"\nPress ESC for display";
                ch=getche();
                if (ch=='p')
```

```
                    t.payingcar();
            if (ch=='n')
                    t.nopaycar();
        }while (ch!=27);
        clrscr();
        t.display();
        getch();
}
```

OUTPUT

Press p for paying car
Press n for nopay car
Press ESC for display p

Press p for paying car
Press n for nopay car
Press ESC for display p

Press p for paying car
Press n for nopay car
Press ESC for display p

Press p for paying car
Press n for nopay car
Press ESC for display p

Press p for paying car
Press n for nopay car
Press ESC for display n

Press p for paying car

Press n for nopay car

Press ESC for display n

Press p for paying car

Press n for nopay car

Press ESC for display ESC

Total number of cars: 6

Total ammount collected: Rs.2

Write a function reversit ( ) which passes a string as a parameter and reverses the string using the procedure:
First swap the first and the last character of the string, then the second and the second last character and so on.
Write a program to exercise reversit ( ). The program should get a string from the user. Call reversit ( ) and print out the esult. Use an input method that allows embedded blanks. Test the program with Napoleon's famous phrase, "Able was I ere I saw Elba".

ALGORITHM

1. Start.
2. Declare a function void reversit (char s [ ]).
    I. It receives a string in char s [ ] and reverses it using logical operations.
3. Start the main function.
4. Declare an array s of 30 characters.
5. Get the input of string from the user in s [30].
6. Call reverseit function.
7. Print the result on screen using puts (s).
8. Stop.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<string.h>
void reversit(char s[])
{
        int i,l;
        char temp;
        l=strlen(s);
        l--;
        for (i=0;i<l/2;i++)
        {
                temp=s[i];
                s[i]=s[l-i];
                s[l-i]=temp;
        }
}
void main()
{
        char s[30];
        clrscr();
        cout<<"Enter a string:";
        gets(s);
        reversit(s);
        puts(s);
        getch();
}
```

OUTPUT

Enter a string:Able was I ere I saw Elba

ablE was I ere I saw elbA.

Create a class complex with two integer data members as real and imaginary. Design a method to input the complex number, design a function to perform addition operation between the two complex numbers by overloading the + operator. Finally display the result.
Write a program to test this class.

ALGORITHM

1. Start.
2. Declare a class complex with following in private part.
   I. Two variables re, im of int type.
3. Declare the following in public part.

I. A member function void input ( ).
II. A member function for overloading + operator with one parameter complex operator+ (complex).
III. A member function void display ( ).

4. Close the class.
5. Start the main function.
6. Declare three objects of class as c1, c2, c3.
7. Clear the screen.
8. Call input function through the object c1 of class complex.
9. Again call input function through the object c2 of class complex.
10. Add the two complex numbers using + operator and assign value to c3 c3 = c1 + c2.
11. Call display function through the object c1 of class complex to display first complex number.
12. Call display function through the object c2 of class complex to display second complex number.
13. Call display function through the object c3 of class complex to display resultant complex number.
14. Get the result.
15. Stop.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
class complex
{
        int re,im;
        public:
                void input();
                complex operator+(complex);
                void display();
};
void complex::input()
{
        cout<<"Enter comlex number as real and imaginary part:";
        cin>>re>>im;
}
void complex::display()
{
        cout<<"\nThe complex number is: "<<re<<"+i"<<im;
}
complex complex::operator+(complex c)
{
        complex c1;
        c1.re=re+c.re;
        c1.im=im+c.im;
        return c1;
}
void main()
{
        complex c1,c2,c3;
        clrscr();
        c1.input();
        c2.input();
        c3=c1+c2;
        clrscr();
        c1.display();
        c2.display();
        cout<<"\nThe sum is:\n";
        c3.display();
        getch();
}
```

OUTPUT

Enter comlex number as real and imaginary part:2 3

Enter comlex number as real and imaginary part:4 5

The complex number is: 2+i3

The complex number is: 4+i5

The sum is:

The complex number is: 6+i8

Declare a class cont with a integer static data member cnt to store the number of objects active, a constructor function to increment cnt, a destructor function to decrement cnt, and a static function showcnt( ) to display the value of cnt at that instance.
Write a program to test this class.

ALGORITHM

1. Start.
2. Declare a class cont with following in private part.
   I. A variable cont of static int type.
3. Declare the following in public part.

I. A default constructor cont ( ).
II. A member function static void showcnt ( ).
III. A destructor ~cont ( ).

4. Close the class.
5. Start the main function.
6. Declare a, b as objects of class cont.
7. Clear the screen.
8. Call static function showcnt () of class cont.
9. Start block.
10. Declare c as object of class cont.
11. call static function showcnt () of class cont.
12. Start inner block.
13. Declare d as object of class cont.
14. call static function showcnt () of class cont.
15. Inner block ends.
16. call static function showcnt () of class cont.
17. Block ends.
18. call static function showcnt () of class cont.
19. Stop.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
class cont
{
        static int cnt;
        public:
                cont()
                {
                        cnt++;
                }
                static void showcnt()
                {
                        cout<<"No. of variables active="<<cnt;
                }
                ~cont()
                {
                        cnt--;
                }
};
int cont::cnt;
void main()
{
        cont a,b;
        clrscr();
        cont::showcnt();
        {
                cont c;
                cont::showcnt();
                {
                        cont d;
                        cont::showcnt();
                }
                cont::showcnt();
        }
        cont::showcnt();
}
```

OUTPUT

No. of variables active=2

No. of variables active=3

No. of variables active=4

No. of variables active=3

No. of variables active=2

Consider the following class definition class father{
Protected : int age;
Public :
Father (int x) {age = x;}
Virtual void iam( )
{ cout << "I am the father, my age is :"<<age<<endl;}
};
Derive the two classes son and daughter from the above class and for each, define suitable constructors for these classes.

Now, write a main ( ) that creates objects of the three classes and then call iam( ) for them. Declare pointer to father. Successively, assign addresses of objects of the two derived classes to this pointer and in each case, call iam( ) through the pointer to demonstrate polymorphism in action.

ALGORITHM

1. START.
2. DECLARE A CLASS FATHER WITH DATA MEMBERS age OF INT TYPE AND MEMBER FUNCTION father() AND virtual function iam().
3. DECLARE ANOTHER CLASS SON AND INHERIT IT FROM THE CLASS FATHER.
4. DECLARE ANOTHER CLASS DAUGHTER AND INHERIT IT FROM THE CLASS FATHER..
5. DECLARE AN OBJECT OF CLASS FATHER f.
6. CALL THE FUNCTION iam() OF f.
7. DECLARE AN OBJECT OF CLASS SON s.
8. CALL THE FUNCTION iam() OF s.
9. DECLARE AN OBJECT OF CLASS DAUGHTER d.
10. CALL THE FUNCTION iam() OF d.
11. DECLARE A POINTER OBJECT OF CLASS FATHER f1.
12. STORE THE ADDRESS OF s IN f1.
13. CALL THE FUNCTION iam() OF s THROUGH f1.
14. STORE THE ADDRESS OF s IN f1.
15. CALL THE FUNCTION iam() OF s THROUGH f1.
16. STOP.

PROGRAM

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

class father
{
        protected:int age;

        public:
                father(int x)
                {
                        age=x;
                }
                void virtual iam()
                {
                        cout<<"I AM THE FATHER,my age is :"<<age<<endl;
                }
};

class son:public father
{
        public:
                son(int x,int y):father(x)
                {
                        age=y;
                }

                void iam()
                {
                        cout<<"I AM THE SON,my age is :"<<age<<endl;
                }
};

class daughter:public father
{
        public:
                daughter(int x,int y):father(x)
                {
                        age=y;
                }
```

```
                void iam()
                {
                        cout<<"I AM THE DAUGHTER,my age is :"<<age<<endl;
                }
};

void main()
{
        clrscr();
        father f(50);
        f.iam();
        son s(50,27);
        s.iam();
        daughter d(50,25);
        d.iam();
        father *f1;
        f1=&s;
        f1->iam();
        f1=&d;
        f1->iam();
}
```

OUTPUT

```
I AM THE FATHER,my age is :50
I AM THE SON,my age is :27
I AM THE DAUGHTER,my age is :25
I AM THE SON,my age is :27
I AM THE DAUGHTER,my age is :25
```

A hospital wants to create a database regarding its indoor patients. The information to store include:

a) Name of the patient
b) Date of admission.
c) Disease.
d) Date of discharge.

Create a structure to store the date (year, month and date as its members). Create a base class to store the above information. The member function should include functions to enter information and display a list of all the patients in the database. Create a derived class to store the age f the patients. List the information about all to store the age of patients. List the information about all the pediatric patients (less than twelve years in age).

ALGORITHM

1.START.
2.DECLARE A STRUCTURE WITH VARIABLES INT YEAR, MONTH , DATE.
3.DECLARE A CLASS PATEINT WITH PROTECTED VARIABLES CHAR name[20],
DATE adm , CHAR disease[20] ,DATE dis , PUBLIC FUNCTIONS VOID GETDATA( ) , VOID SHOWDATA ( ).
4.DECLARE ANOTHER CLASS NEW PATIENT INHERITED FROM THE FIRST CLASS PATIENT WITH VARIABLES INT AGE , PUBLIC FUNCTIONS VOID GETAGE ( ) & VOID SHOWINFO ( ).
5 ENTER THE NAME , DATE OF ADMISSION , MONTH OF ADMISSION , YEAR OF THE ADMISSION , DISEASE , DATE OF DISCHARGE , MONTH OF DISCHARGE , YEAR OF THE DISCHARGE OF THE PATIENT .
6. PRINT THE NAME , DATE OF ADMISSION, DISEASE , DATE OF DISCHARGE OF THE PATIENT.
7.STOP.

PROGRAM

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

struct date
{
        int year;
        int month;
        int date;
};

class patient
{
        protected:
          char name[20];
          date adm;
          char disease[20];
          date dis;

        public:

          void getdata();

          void showdata();

};

class new_patient:public patient
{
        int age;

        public:
                void getage();

                void showinfo();

};

void patient::getdata()
{
        cout<<"\nEnter the name of the patient ::";
```

```
        gets(name);
        cout<<"\nEnter the date of addmission of the patient ::";
        cin>>adm.date;
        cout<<"\nEnter the month of addmission of the patient ::";
        cin>>adm.month;
        cout<<"\nEnter the year of addmission of the patient ::";
        cin>>adm.year;
        cout<<"\nEnter the disease of the patient ::";
        gets(disease);
        cout<<"\nEnter the date of discharge of the patient ::";
        cin>>dis.date;
        cout<<"\nEnter the month of discharge of the patient ::";
        cin>>dis.month;
        cout<<"\nEnter the year of discharge of the patient ::";
        cin>>dis.year;
}

void patient::showdata()
{
        cout<<"\nThe name of the patient is:="<<name;
        cout<<"\nThe date of addmission of the patient is:="<<adm.date;
        cout<<"-"<<adm.month;
        cout<<"-"<<adm.year;
        cout<<"\nThe disease of the patient is:="<<disease;
        cout<<"\nThe date of discharge of the patient ::"<<dis.date;
        cout<<"-"<<dis.month;
        cout<<"-"<<dis.year;
}

void new_patient::getage()
{
        cout<<"\nEnter the age of the patient::";
        cin>>age;
}

void new_patient::showinfo()
{
        if(age<12)
        {
                cout<<"\nThe age of the patient is:="<<age;
                showdata();
        }
        else
                cout<<"\nOnly for patients less than 12 in age!";
}
```

```
void main()
{
        clrscr();
        new_patient p;
        p.getdata();
        p.getage();
        p.showinfo();
}
```

OUTPUT

Enter the name of the patient ::JAGMEET

Enter the date of addmission of the patient ::13

Enter the month of addmission of the patient ::3

Enter the year of addmission of the patient ::2005

Enter the disease of the patient ::BROKEN ELBOW

Enter the date of discharge of the patient ::16

Enter the month of discharge of the patient ::3

Enter the year of discharge of the patient ::2005

Enter the age of the patient::10

The age of the patient is:=10
The name of the patient is:=JAGMEET
The date of addmission of the patient is:=13-3-2005
The disease of the patient is:=BROKEN ELBOW
The date of discharge of the patient ::16-3-2005