M.V.M DEGREE COLLEGE OF COMMERCE AND SCIENCE

OBJECT ORIENTED PROGRAMMING

Practical no 1.a

Aim:-Design an employee class for reading and displaying the employee information, the getInfo() and displayInfo() method will be used respectively. where getInfo() will be private method.

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
class employee
{
        int empid;
        char name[30];
        float salary;
        void getInfo();
        {
        cout<<"Enter the employee id";
        cin>>empid;
        cout<<"Enter the employee name";
        cin>>name;
        cout<<"Enter the employee salary";
        cin>>salary;
        }
        public:
        void displayInfo();
        {
        cout<<"The employee id is:"<<empid<<endl;
        cout<<"The employee name is:"<<name<<endl;
        cout<<"The employee salary is:"<<salary<<ensl;
        }
};
```

```
void main()
{
        employee e;
        e.displayInfo();
        getch();
}
```

M.V.M DEGREE COLLEGE OF COMMERCE AND SCIENCE

Practical 1.b

Aim:-Design the class student containing getdata() and displaydata() as two of its methods which will be used for reading and displaying the student information respectively.where getdata() will be private method.

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
class student
{
        int rollno;
        char name[10];
        void getdata()
        {
        cout<<"Enter the student roll no";
        cin>>rollno;
        cout<<"Enter the student name";
        cin>>name;
        }
        public:
        void displaydata()
        {
        getdata();
        cout<<"The student roll no is:"<<rollno<<endl;
        cout<<"The student name is:"<<name<<endl;
        }
};
void main()
{
        student s;
        s.displaydata();
        getch();
```

```
}
```

M.V.M DEGREE COLLEGE OF COMMERCE AND SCIENCE

Practical 1.c

Aim:-Design the class demo which will contain the following methods:- readno(),factorial() for the calculating the factorial of a number,ispalindrome() will check the given number is palindrome,reverseno() will reverse the given number,Armstrong() will check the given number is armstrong or not.where readno() will be private method.

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
#include<math.h>
class demo
{
        private:
        void readno()
        {
        cout<<"Enter the number";
        cin>>num;
        }
        public:
        void factorial();
        void reverseno();
        void ispalindrome();
        void armstrong();
};
void demo:: factorial()
{
        readno();
        int fact=1;
        for(int i=1;i<=num;i++)
        {
        fact=fact*i;
```

```
        }
        cout<<"Factorial="<<fact<<endl;
}
void demo:: reverseno()
{
        readno()
        int n1,d,sum=0;
        n1=num;
        while(n1!=0)
        {
        d=n1%10;
        sum=d+(sum*10);
        n1=n1/10;
        }
        cout<<"Reverse of number="<<sum<<endl;
}
void demo:: ispalindrome()
{
        readno();
        int n1,d,sum=0;
        n1=num;
        while(n1!=0)
        {
        d=n1%10;
        sum=d+(sum*10);
        n1=n1/10;
        }
        if(sum==num)
        cout<<"Number is palindrome"<<endl;
        else
```

```
        cout<<"Number is not palindrome"<<endl;
}
void demo:: armstrong()
{
        readno();
        int n=num,d,sum=0;
        char str[10];
        itoa(n,str,10);
        int len=strlen(str);
        while(n!=0)
        {
        d=n%10;
        sum=sum+pow(d,len);
        n=n/10;
        }
        if(sum==num)
        cout<<"Number is armstrong"<<endl;
        else
        cout<<"Number is not armstrong"<<endl;
}
void main()
{
        clrscr();
        demo d;
        int a;
        cout<<"Select from the following:-
\n1:factorial\n2:reverse\n3:palindrome\n4:armstrong\n\n";
        cin>>a;
        switch(a)
        {
```

```
        case 1:
        {
        d.factorial();
        break;
        }
        case 2:
        {
        d.reverseno();
        break;
        }
        case 3:
        {
        d.ispalindrome();
        break;
        }
        case 4:
        {
        d.armstrong();
        break;
        }
        }
        getch();
}
```

M.V.M DEGREE COLLEGE OF COMMERCE AND SCIENCE

Practical 2.a

Aim:-Write a friend function for adding the two complex number using a single class

```
#include<iostream.h>
#include<conio.h>
class complex
{
        int real,img;
        public:
        void set()
        {
        cout<<"Enter real and imaginary number:";
        cin>>real>>img;
        }
        friend complex add(complex,complex);
        void show();
};
void complex:: show()
{
        cout<<"The sum of complex  number is:"<<real<<"+"<<img<<"i"<<endl;
}
complex add(complex a,complex b)
{
        complex c;
        c.real=a.real+b.real;
        c.img=a.img+b.img;
        return c;
}
void main()
{
```

```
    complex a,b,d;
    a.set();
    b.set();
    d=add(a,b);
    d.show();
    getch();
}
```

M.V.M DEGREE COLLEGE OF COMMERCE AND SCIENCE

Practical 2.b

Aim:-Write a friend function for adding two different distance and display its sum,using two classes

```
#include<iostream.h>
#include<conio.h>
class fi;
class mc
{
        int m,cm;
        public:
        mc()
        {
        m=0;
        cm=0;
        }
        void getdata()
        {
        cout<<"Enter the length in meter and centimeter";
        cin>>m>>cm;
        }
        friend fi add(fi,mc);
};
class fi
{
        int f,i;
        public:
        void getdata()
        {
        cout<<"Enter the length in feet and inch";
```

```
        cin>>f>>i;
        }
        void putdata()
        {
        cout<<f<<"feet"<<i<<"inch";
        }
        friend fi add(fi,mc);
};
fi add(fi f1,mc m1)
{
        fi t;
        float j;
        j=m1.m*100+m1.cm;
        j=j*0.393700787;
        t.f=j/12;
        int k=int(j);
        t.i=k%12;
        t.i=t.i+f1.i;
        t.f=(t.i/12)+t.f+f1.f;
        t.i=t.i%12;
        return(t);
}
void main()
{
        clrscr();
        mc m1;
        fi f1,f2;
        m1.getdata();
        f1.getdata();
        f2=add(f1,m1);
```

```
    cout<<"\nSum length";
    f2.putdata();
    getch();
}
```

Practical 3.a

Aim:-Design a class complex for adding the two complex numbers and also show the use of constructor

```
#include<iostream.h>
#include<conio.h>
class complex
{
        float real,img;
        public:
        void getdata()
        {
        cout<<"\nEnter the real and imaginary number";
        cin>>real>>img;
        }
        void showdata()
        {
        cout<<"\nThe addition of complex number is"<<real<<"+"<<img<<"i\n";
        }
        complex operator+(complex c)
        {
        complex t;
        t.real=real+c.real;
        t.img=img+c.img;
        return t;
        }
        complex()
        {
        cout<<"Constructing\n";
```

```
        }
        ~complex()
        {
        cout<<"Destructing\n";
        }
};
void main()
{
        clrscr();
        complex c1,c2,c3;
        c1.getdata();
        c2.getdata();
        c3=c1+c2;
        c3.showdata();
        getch();
}
```

Practical 3.b.i

Aim:-Design a class geometry containing the methods area() and also overload the area() function

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#define pi 3.14
class geometry
{
        public:
        void area(int);
        void area(int,int);
        void area(float,int,int);
};
void geometry:: area(int a)
{
cout<<"Area of a circle="<<pi*a*a;
}
void geometry:: area(int a, int b)
{
cout<<"Area of a rectangle="<<a*b;
}
void geometry:: area(float t, int a, int b)
{
cout<<"Area of a triangle="<<t*a*b;
}
void main()
```

```
{
int ch
int a,b,r;
clrscr();
geometry g'
cout<<"\n\t Function overloading";
cout<<"\n\t1.Area of a circle\n2.Area of rectangle\n3.Area of a triangle\n4.exit\n";
cout<<"Enter your choice";
cin>>ch;
switch()
{
case 1;
cout<<"Enter radius of the circle";
cin>>r;
g.area(r);
break;
case 2;
cout<<"Enter sides of the rectangle";
cin>>a>>b;
g.area(a,b);
break;
case 3;
cout<<"Enter side of the triangle";
cin>>a>>b;
g.area(0.5,a,b);
break;
case 4;
exit(0);
}
getch();
```

```
}
```

Practical 3.b.ii

Aim: Design a class geometry containing the method volume() and also overload the volume() function.

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#define pi 3.14
class geometry
{
public:
void volume(float,int);
void volume(int,int,int);
void volume(float,int,int);
};
void geometry:: volume(float a,int b)
{
cout<<"Volume of a circle="<<a*pi*b*b*b;
}
void geometry:: volume(int a, int b, int c)
{
cout<<"Volume of a rectangle="<<a*b*c;
}
void geometry:: volume(float a, int b, int c)
{
cout<<"Volume of a triangle="<<a*pi*b*b*c;
}
```

```
void main()
{
int ch
int r,l,b,h;
clrscr();
geometry g’
cout<<”\n\t Function overloading”;
cout<<”\n\t1.Volume of a sphere\n2.Volume of cuboid\n3.Volume of a cone\n4.exit\n”;
cout<<”Enter your choice”;
cin>>ch;
switch()
{
case 1;
cout<<”Enter radius of the sphere”;
cin>>r;
g.volume(1.334,r);
break;
case 2;
cout<<”Enter sides of the cuboid”;
cin>>l>>h>>b;
g.volume(l,b,h);
break;
case 3;
cout<<”Enter side of the cone”;
cin>>r>>h;
g.area(0.334f,r,h);
break;
case 4;
exit(0);
}
```

```
getch();
}
```

Practical 4.a

Aim: Overload the operator unary(-) for demonstrating operator overloading

```
#include<iostream.h>
#include<conio.h>
class subtract
{
int s1,s2,s3;
public:
subtract()
{
s1=s2=s3=0;
}
subtract (int a, int b, int c)
{
s1=a;
s2=b;
s3=c;
}
void display()
{
cout<<s1<<" "<<s2<<" "<<s3;
}
void operator –()
{
s1=-s1;
```

```
s2=-s2;
s3=-s3;
}
};
void main()
{
clrscr();
subtract m1,m2(2,4,6);
-m1;
cout<<"\nobject1:";
m1.display();
-m2;
cout<<"\n\nobject2:";
m2.display();
getch();
}
```

Practical 4.b

Aim: Overload the operator + for adding the timing of two clock. And also pass object as an argument

```
#include<iostream.h>
#include<conio.h>
class time
{
        int h,m,s;
        public:
        time()
        {
        h=m=s=0;
        }
        void gettime();
        void display()
        {
        cout<<h<<":"<<m<<":"<<s;
        }
        time operator +(time);
};
time time:: operator +(time t1)
{
        time t;
        int a=s+t1.s;
        t.s=a%60;
```

```
        int b=(a/60)+m+t1.m;
        t.m=b%60;
        t.h=(b/60)+h+t1.h;
        t.h=t.h%12;
        return t;
}
void time:: gettime()
{
        cout<<"\nEnter the hour:";
        cin>>h;
        cout<<"\nEnter the minute:";
        cin>>m;
        cout<<"\nEnter the second:";
        cin>>s;
}
void main()
{
        clrscr();
        time t1,t2,t3;
        cout<<"\nEnter the first time:";
        t1.gettime();
        cout<<"\nEnter the second time:";
        t2.gettime();
        t3=t1+t2;
        cout<<"\nFirst time:";
        t1.display();
        cout<<"\nSecond time:";
        t2.display();
        cout<<"\nSum of both the time is:";
        t3.display();
```

```
        getch();
}
```

Practical 4.c

Aim: Overload the + operator for concatenating the two string for eg:- "py" + "thon" = python

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<string.h>
class string
{
        char str1{20};
        public:
        void input()
        {
        gets(str1);
        }
        void output()
        {
        cout<<str1<<endl;
        }
        string operator +(string s2)
        {
        string s;
        strcpy(s.str1,"");
        strcat(s.str1,strcat(str1.""));
```

```
        strcat(s.str1,s2.str1);
        return(s);
        }
};
void main()
{
        string s1,s2,s3;
        clrscr();
        cout<<"Enter first string";
        s1.input();
        cout<<"Enter last string";
        s2.input();
        cout<<"first string is";
        s1.output();
        cout<<"Last string is";
        s2.output();
        s3=s1+s2;
        cout<<"Complete string is";
        s3.output();
        getch();
}
```

Practical 5.a

Aim: Design a class for single level inheritance using public and private type derivation

```
#include<iostream.h>
#include<conio.h>
class base
{
        int n1;
        public:
        int n2;
        void getdata();
        int getbase();
        void display();
};
class derived: private base
{
int n3;
public:
void mul();
void show();
};
void base:: getdata()
{
        cout<<"Enter values for n1 and n2";
```

```
        cin>>n1>>n2;
}
int base:: getbase()
{
        return n1;
}
void base:: display()
{
        cout<<"n1="<<n1<<"\n";
}
void derived:: mul()
{
        getdata();
        n3=n2*getbase();
}
void derived:: show()
{
        display();
        cout<<"n2="<<n2<<"\n";
        cout<<"n3="<<n3<<"\n";
}
void main()
{
        derived d;
        d.getdata();
        d.mul();
        d.display();
        d.show();
        d.n2=10;
        d.mul();
```

```
        d.show();
        getch();
}
```

Practical 5.b

Aim: Design a class for multiple inheritance

```
#include<iostream.h>
#include<conio.h>
class teacher
{
        protected:
        int t;
        public:
        void getdata(int);
};
class student
{
        protected:
        int s;
        public:
        void getdata(int);
};
class coordinator: public teacher,public student
{
        public:
        void display();
};
```

```
void teacher:: getdata(int a)
{
        t=a;
}
void student:: getdata(int b)
{
        s=b;
}
void coordinator:: display()
{
        cout<<"t="<<t<<"\n";
        cout<<"s="<<s<<"\n";
        cout<<"t*s="<<t*s<<"\n";
}
void main()
{
        c.getdata(10);
        c.getdata(20);
        c.display();
        getch();
}
```

Practical 5.c

Aim: Implement the hierarchical inheritance

```
#include<iostream.h>
#include<conio.h>
class Number
{
        public:
        int num;
        void number()
        {
        cout<<"\n\nEnter number=";
        cin>>num;
        }
};
class Square: public Number
{
        public:
        void square()
        {
        number();
        cout<<"\nSquare of the number is"<<(num*num);
        }
};
```

```
class Cube: public Number
{
        public:
        void cube()
        {
        number();
        cout<<"\nCube of the number is"<<(num*num*num);
        }
};
void main()
{
        clrscr();
        Square s;
        s.square();
        Cube c;
        c.cube();
        getch();
}
```

Practical 6.a

Aim: Implement the concept of method overriding

```
#include<iostream.h>
#include<conio.h>
class base
{
        public:
        void display()
        {
        cout<<"Base class\t";
        }
};
class derived: public base
{
        public:
        void display()
        {
        cout<<"derived class";
        }
};
void main()
{
        base b;
```

```
        derived d;
        b.display();
        d.dispaly();
        getch();
}
```

Practical 6.b

Aim: Show the use of virtual function

```
#include<iostream.h>
#include<conio.h>
class one
{
        public:
        int a;
};
class two: virtual public one
{
        public:
        int b;
};
class three: virtual public one
{
        public:
        int c;
};
class four: public two,public three
{
        public:
        int sum;
```

```
};
viod main()
{
        four f;
        clrscr();
        f.a=5;
        f.b=10;
        f.c=15;
        f.sum=f.a+f.b+f.c;
        cout<<"Value of a="<<f.a<<endl;
        cout<<"Value of b="<<f.b<<endl;
        cout<<"Value of c="<<f.c<<endl;
        cout<<"Sum is="<<f.sum<<endl;
        getch();
}
```

Practical 6.c

Aim: Show the immplementation of abstract class

```
#include<iostream.h>
#include<coonio.h>
class shape
{
        protected:
        double width,height;
        public:
        void setdata(double a,double b)
        {
        width=a;
        height=b;
        }
        virtual double area()=0;
};
class rectangle: public shape
{
        public:
        double area()
        {
        return(width*height);
        }
```

```
};
class triangle: public shape
{
        public:
        double area()
        {
        return(width*height)/2;
        }
};
void main()
{
        shape *sPtr;
        rectangle r;
        sPtr=&r;
        sPtr->setdata(2,5);
        cout<<"Area of a rectangle is"<<sPtr->area()<<endl;
        triangle t;
        sPtr=&t;
        sPtr->setdata(2,6);
        cout<<"Area of a triangle is"<<sPtr->area()<<endl;
        getch();
}
```

Practical 7.a

Aim: Write a program to calculate the length of the string

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
void main()
{
        int i,count=0;
        char c[20];
        clrscr();
        cout<<"Enter any string";
        gets;
        for(i=0;c[i]!='\0';i++)
        {
        count++;
        }
        cout<<"String length:"count;
        getch();
}
```

Aim: Write a program to perform string concatenation

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

```
#include<string.h>
void main()
{
        char str1[30],str2[30];
        int i,j;
        cout<<"Enter first string";
        cin>>str1;
        cout<<"Enter second string";
        cin>>str2;
        for(i=0;str1[i]!='\0';++i)
        for(j=0;str2[j]!='\0';++j,++i)
        {
        str1[i]=str2[j];
        }
        str1[i]='\0';
        cout<<"After concatenation"<<str1;
        getch();
}
```

Practical 7.b

Aim: Write a program to perform reverse of a string

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<string.h>
void main()
{
        char str[30],temp;
        int i=0,j=0;
        clrscr();
        cout<<"Enter any string";
        gets(str);
        j=strlen(str)-1;
        while(i<j)
        {
        temp=str[i];
        str[i]=str[j];
        str[j]=temp;
        i++;
        j--;
        }
        cout<<"Reverse of a string is "<<str;
```

```
        getch();
}
```

Aim: Write a program to perform comparision between two string

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
void main()
{
        char a[15],b[15],i,j,flag=0;
        clrscr();
        cout<<"Enter first string";
        gets(a);
        cout<<"Enter second string";
        gets(b);
        i=0;
        j=0;
        while(a[i]!='\0')
        {
        i++;
        }
        while(b[i]!='\0')
        {
        j++;
        }
        if(i!=j)
        {
        flag=0;
        }
        else
```

```
{
for(i=0,j=0;a[i]!='\0',b[j]!='\0';i++,j++)
{
if(a[i]==b[j])
{
flag=1;
}
}
}
if(flag==0)
{
cout<<"String are not equal";
}
else
{
cout<<"String are equal";
}
getch();
}
```

Practical 9.b

Aim: Program to work on multiple files simultaneously

```
#include<iostream.h>
#include<fstream.h>
#include<conio.h>
void main()
{
        ofstream fout;
        fout.open("Book");
        fout<<"Network security\n";
        fout<<"Linux administration\n";
        fout<<"Advance java\n";
        fout.close();
        fout.open("Authors");
        fout<<"Atul kahate\n";
        fout<<"Terry collings\n";
        fout<<"Herbert schildt\n";
        fout.close();
        const int N=50;
        char line[N];
        ifstream.fin;
        fin.open("Book");
        cout<<"Content of book files\n";
```

```
    while(fin)
    {
    fin.getline(line,N);
    cout<<line;
    }
    fin.close();
    fin.open("Authors");
    cout<<"\nContents of Authors file\n";
    while(fin)
    {
    fin.getline(line,N);
    cout<<line;
    }
    fin.close();
    getch();
}
```

Practical 10.b

Aim: Write a program to swap data vary function template

```
#include<iostrem.h>
#include<conio.h>
void swap(T &n1,T &n2)
{
        T temp;
        temp=n1;
        n1=n2;
        n2=temp;
}
void main()
{
        int i1=1,i2=2;
        float f1=1.1,f2=2.2;
        char c1='a',c2='b';
        cout<<"Before passing data to function template\n";
        cout<<"i1="<<i1<<"\ni2"<<i2;
        cout<<"\nf1="<<f1<<"\nf2"<<f2;
        cout<<"\nc1="<<c1<<"\nc2"<<c2;
        swap(i1,i2);
        swap(f1,f2);
        swap(c1,c2);
```

```
    cout<<"After passing data to function template\n";
    cout<<"i1="<<i1<<"\ni2"<<i2;
    cout<<"\nf1="<<f1<<"\nf2"<<f2;
    cout<<"\nc1="<<c1<<"\nc2"<<c2;
    getch();
}
```