OBJECT ORIENTED PROGRAMMING

(OOPs)

LAB MANUAL

INDEX

IT-206 E C ++ Programming Lab.

L T P   Class Work: 25
- - 2   Exam: 25
Total: 50
Duration of Exam: 3 Hrs.

1. Raising a number n to a power p is the 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, and 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 a main ( ) function that gets values from the user to test this function.

2. A point on the two dimensional plane can be represented by two numbers: an X coordinate and a Y coordinate. For example, (4,5) represents a point 4 units to the right of the origin along the X axis and 5 units up the Y axis. The sum of two points can be defined as a new point whose X coordinate is the sum of the X coordinates of the points and whose Y coordinate is the sum of their Y coordinates.
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. Than set the third point equal to the sum of the other two,
and display the value of the new point. Interaction with the program might look like this:
Enter coordinates for P1: 3 4
Enter coordinates for P2: 5 7
Coordinates of P1 + P2 are : 8, 11

3. Create the equivalent of a 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, subtracting, 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 . Some sample interaction with the program might look like this.
Enter first number, operator, second number: 10/ 3
Answer = 3.333333
Do another (Y/ N)? Y
Enter first number, operator, second number 12 + 100
Answer = 112
Do another (Y/ N) ? N

4. 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 structure to store these three parts of a phone number separately. Call the structure phone. Create two structure variables of type phone. Initialize one, and have the user input a number for the other one. Then display both numbers. The interchange might look like this:

```
Enter your area code, exchange, and number: 415 555 1212
My number is (212) 767-8900
Your number is (415) 555-1212
```

5. Create two classes DM and DB which store the value of distances. DM stores
distances in metres 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
maybe
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 metres and cenitmetres depending
on
the object on display.

Q 6. Create a class rational which represents a numerical value by two double values-
NUMERATOR & DENOMINATOR. Include the following public member
Functions:
constructor with no arguments (default).
constructor with two arguments.
void reduce( ) that reduces the rational number by eliminating the highest
common factor between the numerator and denominator.
Overload + operator to add two rational number.
Overload >> operator to enable input through cin.
Overload << operator to enable output through cout.
Write a main ( ) to test all the functions in the class.

Q 7. 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<< end1:}
};
```

Derive the two classes son and daughter from the above class and for each, define iam ( ) to
write
our similar but appropriate messages. You should also define suitable constructors for these
classes.
Now, write a main ( ) that creates objects of the three classes and then calls 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.

Q 8. 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.

Q9. 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 of the
patients. List the information about all the to store the age of the patients. List the information about all the pediatric patients (less than twelve years in age).

Q 10. 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 superclass object. Supply a test program that tests these classes and methods.

Q11. Imagine a tollbooth with a class called toll Booth. The two data items are a type 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 payingCar ( ) increments the car total and adds 0.50 to the cash total. Another function, called nopayCar ( ), 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 key to count
a paying car, and another to count a nonpaying car. Pushing the ESC kay should cause the program to print out the total cars and total cash and then exit.

Q12. Write a function called reversit ( ) that reverses a string (an array of char). Use a for loop
that swaps the first and last characters, then the second and next to last characters and so on. The string should be passed to reversit ( ) as an argument.
Write a program to exercise reversit ( ). The program should get a string from the user, call reversit ( ), and print out the result. Use an input method that allows embedded blanks. Test the
program with Napoleon s famous phrase, Able was I ere I saw Elba) .

Q13. Create some objects of the string class, and put them in a Deque-some at the head of the Deque and some at the tail. Display the contents of the Deque using the forEach ( ) function and a user written display function. Then search the Deque for a particular string, using the first That ( ) function and display any strings that match. Finally remove all the items from the Deque using the getLeft ( ) function and display each item.

Notice the order in which the items are displayed: Using getLeft ( ), those inserted on
the left (head) of the Deque are removed in last in first out order while those put
on the right side are removed in first in first out order. The opposite would be true
if getRight ( ) were used.

Q 14. Create a base class called shape. Use this class to store two double type values that could
be used to compute the area of figures. Derive two specific classes called triangle and
rectangle from the base shape. Add to the base class, a member function get_data ( ) to
initialize base class data members and another member function display_area ( ) to
compute and display the area of figures. Make display_area ( ) as a virtual function and
redefine this function in the derived classes to suit their requirements.Using these three
classes, design a program that will accept dimensions of a triangle or a rectangle
interactively and display the area.
Remember the two values given as input will be treated as lengths of two sides in the case of
rectangles and as base and height in the case of triangles and used as follows:
Area of rectangle = x * y
Area of triangle = ½ * x * y

Rationale BehindObject-oriented programming

Object-oriented programming (OOP) is a programming paradigm that uses abstraction to create models based on the real world. It utilizes several techniques from previously established paradigms, including modularity, polymorphism, and encapsulation. Even though it originated in the 1960s, OOP was not commonly used in mainstream software application development until the 1990s. Today, many popular programming languages (such as Java, JavaScript, C#, C++, Python) support OOP.

Object-oriented programming's roots reach all the way back to the creation of the Simula programming language in the 1960s, when the nascent field of software engineering had begun to discuss the idea of a software crisis. As hardware and software became increasingly complex, how could software quality be maintained? Object-oriented programming in part addresses this problem by strongly emphasizing modularity in software.

Object-oriented programming may be seen as a collection of cooperating *objects*, as opposed to a traditional view in which a program may be seen as a collection of functions, or simply as a list of instructions to the computer. In OOP, each object is capable of receiving messages, processing data, and sending messages to other objects. Each object can be viewed as an independent little machine with a distinct role or responsibility.

Object-oriented programming is intended to promote greater flexibility and maintainability in programming, and is widely popular in large-scale software engineering. By virtue of its strong emphasis on modularity, object oriented code is intended to be simpler to develop and easier to understand later on, lending itself to more direct analysis, coding, and understanding of complex situations and procedures than less modular programming methods.

HARDWARE REQUIREMENT

- PENTIUM I   (200 GHZ)
- 128 MB RAM
- 20 GB HDD

SOFTWARE REQUIREMENT

- WINDOWS 9x or above
- Language C ++

PROGRAM 1 :-Find the power of the number

```
#include<iostream.h>
#include<conio.h>
double power(double n,int p=2);
int main()
{
double n,r;
int p;
char c;
clrscr();
cout<<"enter the number:";
cin>>n;
do
        {
        cout<<"do you want to enter power(y/n)?:";
        cin>>c;
        if(c=='y')
                {
                cout<<"enter the power to be raised:";
                cin>>p;
                r=power(n,p);
                }
        else
                {
                if(c=='n')
                        {
                        p=2;
                        r=power(n);
                        }
                else
                        cout<<"invalid choice\n";
                }
        }while(c!='y'&&c!='n');
cout<<n<<"^"<<p<<"("<<n<<" raised to the power "<<p<<")="<<r;
getch();
return 0;
}
double power(double n,int p)
{
double r=1;
int i;
if(p<0)
        r=1/power(n,-p);
else
        for(i=1;i<=p;i++)
```

```
                    r=r*n;
return(r);
}
```

OUTPUT:

```
enter the number:4
do you want to enter power(y/n)?:y
enter the power to be raised:3
4^3(4 raised to the power 3)=6
```

PROGRAM 2: Write a program that uses co-ordinate to model a point

```
#include<iostream.h>
#include<conio.h>
#define N 2
struct point
{
int x;
int y;
}p[N],pt={0,0};
int main()
{
int i;
clrscr();
for(i=0;i<=N-1;i++)
        {
        cout<<"enter coordinates x"<<i+1<<" & y"<<i+1<<":";
        cin>>p[i].x>>p[i].y;
        }
for(i=0;i<=N-1;i++)
        {
        pt.x=pt.x+p[i].x;
        pt.y=pt.y+p[i].y;
        }
cout<<"sum of "<<N<<" points is:"<<pt.x<<","<<pt.y;
getch();
return 0;
}
```

OUTPUT:

```
enter coordinates x1 & y1:2 4
enter coordinates x2 & y2:3 7
sum of 2 points is:5,11
```

PROGRAME 3: Write a program to make a Calculator

```
#include<iostream.h>
#include<conio.h>
int main()
{
char c,operatr;
double operand1,operand2;
clrscr();
do
        {
        cout<<"Enter first number,operator,second number:";
        cin>>operand1>>operatr>>operand2;
        switch(operatr)
                {
                case '+':cout<<"Answer="<<operand1+operand2;
                break;
                case '-':cout<<"Answer="<<operand1-operand2;
                break;
                case '*':cout<<"Answer="<<operand1*operand2;
                break;
                case '/':cout<<"Answer="<<operand1/operand2;
                break;
                default:cout<<"invalid operator";
                }
        do
                {
                cout<<"\nDo another(y/n)?:";
                cin>>c;
                if(c!='y'&&c!='n')
                        cout<<"invalid choice";
                }while(c!='y'&&c!='n');
        }while(c=='y');
return 0;
}
```

OUTPUT:

```
Enter first number,operator,second number:2+3
Answer=5
Do another(y/n)?:y
Enter first number,operator,second number:4*8
Answer=32
Do another(y/n)?:y9/3
Enter first number,operator,second number:Answer=3
Do another(y/n)?:n
```

Write a

PROGRAM 4. Write a program that uses a structure to store the three parts of a phone number separately.

```
#include<iostream.h>
#include<conio.h>
struct phone
{
char area[10];
char exchange[10];
char number[10];
};
int main()
{
phone ph1={"212","767","8900"};
phone ph2;
clrscr();
cout<<"\nenter your area code,exchange and number:";
cin>>ph2.area>>ph2.exchange>>ph2.number;
cout<<"\nmy number is ("<<ph1.area<<")"<<ph1.exchange<<"-"<<ph1.number;
cout<<"\nyour number is ("<<ph2.area<<")"<<ph2.exchange<<"-"<<ph2.number;
getch();
return 0;
}
```

OUTPUT:

```
enter your area code,exchange and number:011 257 3333

my number is (212)767-8900
your number is (011)257-3333
```

PROGRAM 5: Create two classes DM and DB which store the value of distances.

```
#include<iostream.h>
#include<conio.h>
class DB;
class DM
{
int metres;
float centimetres;
public:
DM()
{
metres=0;centimetres=0.00;
}
DM(int m,float cm)
{
metres=m;
centimetres=cm;
}
int get_m()
{
return metres;
}
float get_cm()
{
return centimetres;
}
void getdata()
{
cout<<"enter metres:";
cin>>metres;
cout<<"enter centimetres:";
cin>>centimetres;
}
void display()
{
cout<<metres<<"m-"<<centimetres<<"cm";
}
friend DM add(DM,DB);
};
class DB
{
int feet;
float inches;
public:
DB()
```

```
{
feet=0;
inches=0.00;
}
DB(int f,float i)
{
feet=f;
inches=i;
}
DB(DM dm)
{
int m;
float cm,t_cm,t_in;
m=dm.get_m();
cm=dm.get_cm();
t_cm=m*100+cm;
t_in=t_cm/2.54;
feet=int(t_in)/12;
inches=t_in-feet*12;
}
operator DM()
{
float in=feet*12+inches;
float cm=in*2.54;
int mtr=int(cm)/100;
float cmtr=cm-mtr*100;
return DM(mtr,cmtr);
}
void getdata()
{
cout<<"enter feet:";
cin>>feet;
cout<<"enter inches:";
cin>>inches;
}
void display()
{
cout<<feet<<"\'-"<<inches<<"\"";
}
friend DM add(DM,DB);
};
DM add(DM dm,DB db)
{
DM a=db;
int m=dm.metres+a.metres;
float cm=dm.centimetres+a.centimetres;
```

```
if(int(cm)>=100)
        {
        cm-=100.00;
        m++;
        }
return DM(m,cm);
}
int main()
{
DB db,db1;
DM dm,dm1;
clrscr();
cout<<"enter distance d1(in metres & centimetres):\n";
dm.getdata();
cout<<"enter distance d2(in feet & inches):\n";
db.getdata();
dm1=add(dm,db);
db1=add(dm,db);
dm.display();cout<<" + ";db.display();cout<<" = ";dm1.display();
cout<<" = ";
db1.display();
getch();
return 0;
}
```

<u>OUTPUT:</u>

```
enter distance d1(in metres & centimetres):
enter metres:2
enter centimetres:22
enter distance d2(in feet & inches):
enter feet:12
enter inches:12
2m-22cm + 12'-12" = 6m-18.23999cm = 20'-3.401566"
```

<u>POGRAM 6</u>: Create a class rational which represents a numerical value by two double values-

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
class rational
{
private:
int num;
int den;
int lcm(int a,int b);
int hcf(int a,int b);
public:
rational()
{
num=0;
den=1;
}
rational(int n,int d=1)
{
if(d==0)
        {
        cout<<"denominator can't be zero";
        getch();
        exit(1);
        }
else
        {
        num=n;
        den=d;
        }
}
rational reduce()
{
rational temp;
int h=hcf(num,den);
temp.num=num/h;
temp.den=den/h;
return temp;
}
rational operator +(rational r)
{
int l,t1,t2,x1,x2,den1,den2,p;
rational ans;
if(den<0)
        den1=-den;
```

```
else
        den1=den;
if(r.den<0)
        den2=-r.den;
else
        den2=r.den;
p=lcm(den1,den2);
if((den<0&&r.den>0)||(den>0&&r.den<0))
        l=-p;
else
        l=p;
ans.den=l;
t1=l/den;
t2=l/r.den;
x1=num*t1;
x2=r.num*t2;
ans.num=x1+x2;
return ans;
}
friend istream& operator >> (istream& s,rational& r);
friend ostream& operator << (ostream& s,rational& r);
};
int rational::lcm(int a,int b)
{
int i,lcm=1;
while(!(a==1&&b==1))
        {
        i=2;
        while(!(a%i==0||b%i==0)&&i<(a>b?a:b))
                i++;
        lcm*=i;
        if(a%i==0)
                a/=i;
        if(b%i==0)
                b/=i;
        }
return lcm;
}
int rational::hcf(int a,int b)
{
int r=a%b;
while(r)
        {
        a=b;
        b=r;
        r=a%b;
        }
return b;
}
istream& operator >>(istream& s,rational& r)
{
int a,b;
char c;
s>>a>>c>>b;
if(c!='/')
        {
        cout<<"use of invalid notation";
        getch();
        exit(0);
        }
if(b==0)
        {
        cout<<"denominator can't be zero.";
        getch();
        exit(1);
        }
r.num=a;
r.den=b;
return s;
}
ostream& operator <<(ostream& s,rational& r)
{
if(r.den==1)
        s<<r.num;
else
        {
        if(r.den==-1)
                s<<-r.num;
        else
                s<<r.num<<'/'<<r.den;
        }
return s;
}
int main()
{
clrscr();
rational r1,r2,r3;
cout<<"enter r1:";
cin>>r1;
cout<<"enter r2:";
cin>>r2;
r3=r1+r2;
```

```
cout<<r1<<" + "<<r2<<" = "<<r3<<" = "<<r3.reduce();
getch();
return 0;
}
```

OUTPUT:

```
enter r1:2/3
enter r2:4/5
2/3 + 4/5 = 22/15 = 22/15
```

PROGRAM 7: Derive the two classes son and daughter, and demonstrate polymorphism in action.

```
#include<iostream.h>
#include<conio.h>
class father
{
protected:unsigned int age;
public:
father()
{
age=60;
}
father(int x)
{
age=x;
}
virtual void iam()
{
cout<<"I AM THE FATHER,my age is:"<<age<<endl;
}
};
class son:public father
{
public:
son()
{
age=30;
}
son(int x)
{
age=x;
}
void iam()
{
cout<<"I AM THE SON,my age is:"<<age<<endl;
}
};
class daughter:public father
{
public:
daughter()
{
age=24;
}
daughter(int x)
{
```

```
age=x;
}
void iam()
{
cout<<"I AM THE DAUGHTER,my age is:"<<age<<endl;
}
};
int main()
{
father f(50),*ptrf;
son s(23);
daughter d(16);
clrscr();
cout<<"\n\n******************************CALL BY FATHER
OBJECT*************\
****************\n\n";
f.iam();
cout<<"\n\n*******************************CALL BY SON
OBJECT***************\
****************\n\n";
s.iam();
cout<<"\n\n*****************************CALL BY DAUGHTER
OBJECT************\
****************\n\n";
d.iam();
ptrf=&s;
cout<<"\n\n***************CALL BY POINTER TO FATHER WITH ADDRESS
OF\
 SON OBJECT*************\n\n";
ptrf->iam();//(*ptrf).iam();
cout<<"\n\n************CALL BY POINTER TO FATHER WITH ADDRESS OF\
 DAUGHTER OBJECT***********\n\n";
ptrf=&d;
ptrf->iam();//(*ptrf).iam();
cout<<"\n\n============================================================
==========\
=================";
getch();
return 0;
}
```

<u>OUTPUT:</u>

```
******************************CALL BY FATHER
OBJECT*****************************

I AM THE FATHER,my age is:50

*******************************CALL BY SON
OBJECT*******************************

I AM THE SON,my age is:23

*****************************CALL BY DAUGHTER
OBJECT****************************

I AM THE DAUGHTER,my age is:16

***************CALL BY POINTER TO FATHER WITH ADDRESS OF SON
OBJECT*************

I AM THE SON,my age is:23

************CALL BY POINTER TO FATHER WITH ADDRESS OF DAUGHTER
OBJECT***********

I AM THE DAUGHTER,my age is:16
```

<u>PROGRAM 8</u>: Create a database regarding its indoor patients.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
struct DATE
{
int date;
int month;
int year;
};
class patient
{
protected:
char name[40];
DATE admit;
char disease[40];
DATE discharge;
public:
void enter()
{
char c;
int i;
cout<<"Enter information:"<<endl;
cout<<"Name:";
for(i=0;i<=38;i++)
{
cin>>c;
if(c=='\r')
break;
else
name[i]=c;
}
name[i]='\0';
//cin.getline(name,40);
//cin.ignore(100,'\n');
cout<<"Date of admission:"<<endl;
cout<<"            Date:";
cin>>admit.date;
cout<<"            Month:";
cin>>admit.month;
cout<<"            Year:";
cin>>admit.year;
//cin.ignore(100,'\n');
cout<<"Disease:";
for(i=0;i<=38;i++)
{
```

```
cin>>c;
if(c=='\r')
break;
else
disease[i]=c;
}
disease[i]='\0';
//cin.getline(disease,40);
//cin.ignore(100,'\n');
cout<<"Date of discharge:"<<endl;
cout<<"          Date:";
cin>>discharge.date;
cout<<"          Month:";
cin>>discharge.month;
cout<<"          Year:";
cin>>discharge.year;
}
void display()
{
cout<<setw(14);
cout.setf(ios::left);
cout<<name<<setw(12);
cout.unsetf(ios::left);
cout<<admit.date<<'-'<<admit.month<<'-'<<admit.year;
cout<<setw(17);
cout<<disease<<setw(13);
cout<<discharge.date<<'-'<<discharge.month<<'-'<<discharge.year;
}
};
class pat_age:public patient
{
int age;
public:
int get_age()
{
return age;
}
void enter()
{
patient::enter();
cout<<"Age:";
cin>>age;
}
void display()
{
patient::display();
cout<<setw(6)<<age;
}
};
```

PROGRAM 9. Find the total no of cars.

```
include<iostream.h>
#include<conio.h>
const char ESC=27;
const double TOLL=0.5;
class tollbooth
{
private:
unsigned int totalcars;
double totalcash;
public:
tollbooth()
{
totalcars=0;
totalcash=0;
}
void payingcar()
{
totalcars+=1;
totalcash+=TOLL;
}
void nopaycar()
{
totalcars+=1;
}
void display()
{
cout<<"\ncars="<<totalcars<<",cash="<<totalcash;
}
};
int main()
{
tollbooth booth1;
char ch;
clrscr();
cout<<"\npress 0 for each non-paying car,"
   <<"\n      1 for each paying car,"
   <<"\n      Esc to exit the program.\n";
do
        {
        ch=getche();
        if(ch=='0')
                booth1.nopaycar();
        if(ch=='1')
```

```
                    booth1.payingcar();
          }while(ch!=ESC);
booth1.display();
getch();
return 0;
}
```

OUTPUT:

```
press 0 for each non-paying car,
     1 for each paying car,
     Esc to exit the program.
←222
cars=1,cash=0.5
```

PROGRAM 10 Write a Program to   reversea string

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
const int MAX=80;
void reversit(char[]);
int main()
{
char str[MAX];
clrscr();
cout<<"\nenter the string:";
cin.get(str,MAX);
reversit(str);
cout<<"reversed string is:";
cout<<str;
getch();
return 0;
}
void reversit(char s[])
{
int len=strlen(s);
for(int j=0;j<len/2;j++)
        {
        char temp=s[j];
        s[j]=s[len-j-1];
        s[len-j-1]=temp;
        }
}
```

OUTPUT:

```
enter the string:sdhgd
reversed string is:dghds
```

PROGRAM 11: Create a base class called shape. Use this class to store two double type values that could

```
#include<iostream.h>
#include<conio.h>
class shape
{
protected:
double x;
double y;
public:
shape()
{
x=0.0;
y=0.0;
}
shape(double a,double b)
{
x=a;
y=b;
}
void get_data()
{
cout<<"enter x:";
cin>>x;
cout<<"enter y:";
cin>>y;
}
virtual void display_area()=0;
};
class triangle:public shape
{
public:
triangle()
{}
triangle(double a,double b):shape(a,b)
{}
void display_area()
{
cout<<"Area of triangle:"<<(x*y)/2;
}
};
class rectangle:public shape
```

```
{
public:
rectangle()
{}
rectangle(double a,double b):shape(a,b)
{}
void display_area()
{
cout<<"Area of rectangle:"<<x*y;
}
};
int main()
{
int c;
clrscr();
while(1)
        {
        cout<<"\n1.area of triangle";
        cout<<"\n2.area of rectangle";
        cout<<"\n3.exit";
        cout<<"\nenter your choice:";
        cin>>c;
        switch(c)
                {
                case 1:triangle t;
                     t.get_data();
                     t.display_area();
                     break;
                case 2:rectangle r;
                     r.get_data();
                     r.display_area();
                     break;
                case 3:return 0;
                default:cout<<"invalid choice";
                }
        }
}
```

<u>OUTPUT:</u>

```
1.area of triangle
2.area of rectangle
3.exit
enter your choice:2
enter x:2
enter y:5
Area of rectangle:10
1.area of triangle
2.area of rectangle
3.exit
enter your choice:3
```

.

<u>PROGRAM 12 :</u> . Create some objects of the string class, and put them in a Deque-some at the head of the. Deque and some at the tail.

```
#include<iostream.h>
#include<conio.h>
#include<strng.h>
#include<deque.h>
void actionfunc(Object&,void*);
int testfunc(const Object& obj,void*);
int main()
{
Deque deq;
String s1("Rajtilak");
String s2("Deepak");
String s3("Lakshay");
String s4("Rishi");
String s5("Amit");
clrscr();
deq.putLeft(s1);
deq.putLeft(s2);
deq.putLeft(s3);
deq.putRight(s4);
deq.putRight(s5);
cout<<"\n\nIterate through deque:";
deq.forEach(actionfunc,0);
String& temp=(String&)deq.firstThat(testfunc,0);
if(temp!=NOOBJECT)
        {
        cout<<"\n\nMatch with:";
        temp.printOn(cout);
        }
else
        cout<<"\n\nNo match.";
cout<<"\n\nDisplay and remove items from deque:";
while(!deq.isEmpty())
        {
        String& temp=(String&)deq.getLeft();
        cout<<endl;
        temp.printOn(cout);
        }
cout<<"\nContents of empty deque:";
deq.printOn(cout);
getch();
return 0;
}
void actionfunc(Object& obj,void*)
{
cout<<endl;
obj.printOn(cout);
}
int testfunc(const Object& obj,void*)
{
String& temp=(String&)obj;
String test("Deepak");
if(temp.isEqual(test))
        return 1;
else
        return 0;
}
```

PROGRAM 13: . Make a class Employee with a name and salary

```
#include<iostream.h>
#include<conio.h>
#include<strng.h>
class employee
{
protected:
char name[40];
unsigned int salary;
};
class manager:public employee
{
String department;
public:
manager(char n[],unsigned int s,String d)
{
strcpy(name,n);
salary=s;
department=d;
}
void tostring()
{
cout<<"Name:"<<name<<endl<<"Deptt:"<<department<<endl<<"Salary:"<<"Rs."
   <<salary<<endl;
}
};
class executive:public manager
{
public:
executive(char n[],unsigned int s,String d):manager(n,s,d)
{
}
void tostring()
{
String s("Executive");
s.printOn(cout);
cout<<endl;
manager::tostring();
}
};
int main()
{
```

```
String d1("Information Technology");
String d2("Mechanical");
manager m("Rajtilak Bhatt",15000,d1);
executive e("Deepak Bhatt",21000,d2);
clrscr();
cout<<"\n\n*****************************CALL BY MANAGER
OBJECT*************\
****************\n\n";
m.tostring();
cout<<"\n\n******************CALL BY EXECUTIVE OBJECT DERIVED FROM
MANAGER*\
****************\n\n";
e.tostring();
cout<<"\n\n========================================================
=========\
================";
getch();
return 0;
}
```

REFERENCES

Text Books:

- C++ How to Program by **H M Deitel** and **P J Deitel**.
- Object Oriented Programming in Turbo C++ by **Robert Lafore** , Press.
- Programming with C++ By **D Ravichandran**,

Reference books:

- Object oriented Programming with C++ by **E Balagurusamy**.
- Computing Concepts with C++ Essentials by **Horstmann**.
- The Complete Reference in C++ **By Herbert Schildt**.

NEW IDEAS BESIDES UNIVERSITY SYLLABUS

Apart from Implementing the list of practicals given in the Syllabus.Students are given 8-10 extra practicals to perform . Also to improve the fundamentals in C++ the student are given a mini project which is assessed in the end of the semester. This updates them and clears their logic on the mother of all languages i.e C++.

Doubts are cleared in the tutorials and in Lab hours . Extra lab hours are provided to the students ,all these initiative enhances the skills of the students ..

FAQs in OOPS

- SECTION 1: BASICS
  - 1.1) What Is an Object?
  - 1.2) What Is Object Encapsulation (Or Protection)?
  - 1.3 What Is A Class?
  - 1.4) What Is a Meta-Class?
  - 1.5) What Is the Infinite Regress Of Objects And Classes?
  - 1.6) What Are MOPs and Reflection?
  - 1.7) What Is Inheritance?
  - 1.8) What Is Multiple Inheritance?
  - 1.9) Does Multiple Inheritance Pose Any Additional Difficulties?
  - 1.10) What Is Dynamic Inheritance?
  - 1.11) What Is Shared (Repeated) Inheritance?
  - 1.12) Why Use Inheritance?
  - 1.13) Why Don't Some People Like Inheritance?
  - 1.14) What Is Specialization/Generalization/Overriding?
  - 1.15) What Is the Difference between Object-Based and Object-Oriented?
  - 1.16) Is A Class An Object?
  - 1.17) Is an Object A Class?
  - 1.18) What Is a Method? (And Receiver And Message)
  - 1.19) What Are Multi-Methods and Multiple-Polymorphism?
  - 1.20) What Is OOP?
  - 1.21) What Is OOA/OOD (And Where Can I Get What I Need On It)?
  - 1.22) Where Did Object-Orientation Come From?
  - 1.23) What Are the Benefits of Object-Orientation?

SECTION 2:

- 2.1) What Is Polymorphism?
- 2.2) How does Polymorphism Boil Down In OO Programming Languages?
- 2.3) What Is Dynamic Binding?
- 2.4) Is There A Difference Between Being A Member Or Instance Of A Class?
- 2.5) What Is The Difference Between Static And Dynamic Typing?
- 2.6) What Is A difference Between Type And Class (Representation)?
- 2.7) What Are Generics and Templates?

SECTION 3:

NEW IDEAS BESIDES UNIVERSITY SYLLABUS

Apart from Implementing the list of practicals given in the Syllabus.Students will be given 8-10 extra practicals to perform . Also to improve the fundamentals in C++ the student will be given a mini project which will be assessed the end of the semester. This will update them on the motherof all languages i.e C++.

Doubts will be be clearedin the tutorials andin Lab hours . Extralab horscan be providedto the students ,all theseiniciative will enhance the skills of the students ..