**Lab – 6**

**Classification of Polymorphism, Compile time and Run time Polymorphism, Pointers to**

**derived class object, Virtual functions, Pure virtual functions.**

**1. Class polygon contains data member width and height and public method set\_value()**

**to assign values to width and height. Class Rectangle and Triangle are inherited from**

**polygon class. Both the classes contain public method calculate\_area() to calculate the**

**area of Rectangle and Triangle. Use base class pointer to access the derived class**

**object and show the area calculated.**

//base class ptr to derived class

#include<iostream>

using namespace std;

class polygon{

protected:

int width;

int height;

public:

void set\_val(int a, int b){

width=a;

height=b;

}

virtual void calculate\_area(){}; //key learning- to change behaviour of pointer

};

class rect: public polygon{

public:

void calculate\_area(){

cout<<"Area is: "<<width\*height<<endl;

}

};

class triangle: public polygon{

float area;

public:

void calculate\_area(){

area=(1/2.0)\*width\*height;

cout<<"Area is: "<<area;

}

};

int main()

{

polygon \*ptr;

rect obj1;

obj1.set\_val(2,2);

ptr=&obj1;

ptr->calculate\_area();

triangle obj2;

obj2.set\_val(2,2);

ptr=&obj2;

//ptr->set\_val(2,2)

ptr->calculate\_area();

delete ptr;

return 0;

}

**2. Write a program to create a class shape with functions area and display to find area**

**and display the name of the shape and other essential component of the class. Create**

**derived classes circle, rectangle and triangle each having overridden functions area**

**and display. Write a program to find and display the area of circle, rectangle and**

**triangle.**

//overriding functions

#include<iostream>

using namespace std;

//pure virtual in base

class shape{

public:

virtual void area(){

};

virtual void display(){};

};

class circle: public shape{

float area1;

int r;

public:

void get\_val(){

cout<<"Enter r: "<<endl;

cin>>r;

}

void area(){

area1=(3.14)\*r\*r;

}

void display(){

cout<<"Area is: "<<area1<<endl;

}

};

class rect: public shape{

float area1;

int width;

int height;

public:

void get\_val(){

cout<<"Enter w and h: "<<endl;

cin>>width>>height;

}

void area(){

area1=width\*height;

}

void display(){

cout<<"Area is: "<<area1<<endl;

}

};

class triangle: public shape{

float area1;

int width;

int height;

public:

void get\_val(){

cout<<"Enter w and h: "<<endl;

cin>>width>>height;

}

void area(){

area1=(1/2.0)\*width\*height;

}

void display(){

cout<<"Area is: "<<area1;

}

};

int main()

{

circle obj1;

obj1.get\_val();

obj1.area();

obj1.display();

rect obj2;

obj2.get\_val();

obj2.area();

obj2.display();

triangle obj3;

obj3.get\_val();

obj3.area();

obj3.display();

return 0;

}

**3. Write a C++ program to compute area of right angle triangle, equilateral triangle,**

**isosceles triangle using function overloading concept.**

#include<iostream>

#include<math.h>

using namespace std;

class AREA{

public:

void area(int b,int h){

cout<<"area of right triangle is: "<<0.5\*b\*h<<endl;

}

void area(int a){

cout<<"Area of eq triangle is: "<<sqrt(3)\*a\*a\*0.5<<endl;

}

void area(int a, int b, int c){

cout<<"area of iso triangle is: "<<0.5\*a\*sqrt((b\*b)-c\*c)<<endl;

}

};

int main(){

AREA a;

a.area(2,4);

a.area(1);

a.area(10,5,1);

}

**4. Write a program with Student as abstract class and create derive classes Engineering,**

**Medicine and Science from base class Student. Create the objects of the derived**

**classes and process them and access them using array of pointer of type base class**

**Student.**

#include<iostream>

using namespace std;

class student{

public:

virtual void display(){

};

};

class science: public student{

public:

void display(){

cout<<"Hi Science"<<endl;

}

};

class engineering: public student{

public:

void display(){

cout<<"Hi Engineering"<<endl;

}

};

class medicine: public student{

public:

void display(){

cout<<"Hi Medicine"<<endl;

}

};

int main(){

student \*ptr[3];

ptr[0]=new science;

ptr[0]->display();

ptr[1]=new engineering;

ptr[1]->display();

ptr[2]=new medicine;

ptr[2]->display();

delete \*ptr;

return 0;

}