```cpp
/*
 Program: Shape
 Author: Michael Campbell
 Date: 5/11/2011
 Synopsis: Final Q#2
 */

#include <iostream>
#include "Circle.h"
#include "Rectangle.h"
#include "Square.h"
#include "Triangle.h"
using namespace std;
using namespace Shape;

int main ()
{
    float distance = 0;

    Circle * CIRC;
    CIRC = new Circle[3];

    int height = 4;
    int width = 6;
    Circle circ;
    circ.setSize(3.0);
    circ.setColor(Indigo);
    circ.setCenter(-10, -30);
    CIRC[0] = circ;
    loop(circ);

    Circle circ2(10, 0, 0, Yellow);
    Circle circ3(16, 16, 16, Red);

    Rectangle rect(width, height);
    CIRC[1] = circ2;
    CIRC[2] = circ3;

    //SortShape(CIRC, 3);

    loop(rect);
    Square sqr(4);

    loop(sqr);

    distance = getDistance(sqr, circ);
    cout << "The distance between sqr and circ is " << distance << endl;

    Triangle tri(width, height);


    return EXIT_SUCCESS;
}
```

```
//Shape.h
#ifndef SHAPE_H
#define SHAPE_H

namespace Shape {

    enum COLOR {
        Red = 1,
        Orange,
        Yellow,
        Green,
        Blue,
        Indigo,
        Violet
    };

    class Shape {
    protected:
        int X;
        int Y;
        COLOR col;
    public:
        Shape();
        Shape(int xLoc, int yLoc, COLOR color = Red);
        virtual ~Shape();
        void ResetLocation(int deltaX, int deltaY);
        void ResetColor(COLOR color);
        virtual float getArea() = 0;
        virtual void drawObject();
        virtual void getFacts();

        friend float getDistance(Shape &shape1, Shape &shape2);
        friend void SortShape(Shape *array, int NumShapes);
        friend void loop(Shape &shapeS);
    };
}

#endif //SHAPE_H
```

```
//Shape.cpp
#include "Shape.h"
#include "Circle.h"
#include <iostream>
#include <cmath>
using namespace std;

//int X;
//int Y;
//COLOR col;
namespace Shape {

    Shape::Shape():X(0), Y(0), col(Red){
        //blank
    }
    Shape::Shape(int xLoc, int yLoc, COLOR color){
        X = xLoc;
        Y = yLoc;
    }
    Shape::~Shape(){
        //blank
    }
    void Shape::ResetLocation(int deltaX, int deltaY){
        deltaX = 0;
        deltaY = 0;
    }
    void Shape::ResetColor(COLOR color){
        color = Red;
    }
    float Shape::getArea(){
        float Area = 0;
        return Area;
    }
    void Shape::drawObject(){
        cerr << "This should not print";
        EXIT_FAILURE;
    }

    void Shape::getFacts(){
        cerr << "Gets the facts" << endl;
    }

    float getDistance(Shape &shape1, Shape &shape2){
        float Distance = 0;

        Distance = sqrt(((shape2.X - shape1.X)*(shape2.X - shape1.X))+ (shape2.
            Y - shape1.Y)*(shape2.Y - shape1.Y));

        return Distance;
    }
    void SortShape(Circle array[], int NumShapes){
        int i, j, flag = 1;    // set flag to 1 to start first pass
        Circle temp;             // holding variable
//        array = new Circle[NumShapes + 1];
        for(i = 1; (i <= NumShapes) && flag; i++)
        {
            flag = 0;
            for (j=0; j < (NumShapes -1); j++)
            {
                if (array[j+1].Area < array[j].Area)      // ascending order
                    simply changes to <
                {
                    temp = array[j];             // swap elements
                    array[j] = array[j+1];
                    array[j+1] = temp;
                    flag = 1;               // indicates that a swap occurred.
                }
            }
        }
        array->drawObject();
        array->getSize();
        return;
    }

    void loop(Shape &shapeS){
        Shape * ptr;
        *ptr = shapeS;
        float Distance = 0;

        Circle shape2;
        cout << "This shape has coordinates of (" << shapeS.X << ", " << shapeS
            .Y << ")" << endl;
        float getDistance(Shape &shape1, Shape &shape2);
        shapeS.ResetLocation(2, 2);
        shapeS.getFacts();
        shapeS.getArea();
        shapeS.drawObject();
        Distance = getDistance(shapeS, shape2);

    }
}
```

```
//Rectangle.h
#ifndef RECTANGLE_H
#define RECTANGLE_H

#include "Shape.h"

namespace Shape {

    class Rectangle : public Shape{
        int Base;
        int Height;
    public:
        Rectangle():Base(5), Height(10), Shape(15, 15, Orange){}
        Rectangle(int base, int height);
        Rectangle(int base, int height, int x, int y, COLOR color):Base(base),
            Height(height), Shape(x, y, color){}
        ~Rectangle();
        virtual float getArea();
        virtual void drawObject();
        void setSize(int base, int height);
        void getSize();
        void setColor(COLOR color);
        COLOR getColor();
        void setCenter(int xCoord, int yCoord);
        void getCenter();
        virtual void getFacts();
        Rectangle operator +(const Rectangle &rhs);     //add two Rectangles
        float Area;

    };


}


#endif
```

```
//Rectangle.cpp
#include "Rectangle.h"
#include <iostream>
using namespace std;

namespace Shape {
    Rectangle::Rectangle(int base, int height):Base(base), Height(height),
        Shape(15, 15, Orange){
        //blank
    }
    Rectangle::~Rectangle(){

    }
    float Rectangle::getArea(){
        Area = Base * Height;

        return Area;
    }
    void Rectangle::drawObject(){
        cout << "Drawing Rectangle" << endl;
    }
    void Rectangle::setSize(int base, int height){
        Base = base;
        Height = height;
    }
    void Rectangle::getSize(){
        cout << "Base = " << Base << endl;
        cout << "Height = " << Height << endl;

    }
    void Rectangle::setColor(COLOR color){
        col = color;
    }
    COLOR Rectangle::getColor(){
        return col;
    }
    void Rectangle::setCenter(int xCoord, int yCoord){
        X = xCoord;
        Y = yCoord;
    }
    void Rectangle::getCenter(){
        cout << "X = " << X << endl;
        cout << "Y = " << Y << endl;
    }

    void Rectangle::getFacts(){
        cout << "Coordinates are: " << endl;
        cout << "X = " << X << endl;
        cout << "Y = " << Y << endl;
        cout << "Size: " << endl;
        cout << "Base = " << Base << endl;
        cout << "Height = " << Height << endl;
        cout << "Color: " << col << endl;

    }

    Rectangle Rectangle::operator +(const Rectangle &rhs){
        int TotalBase = (Base + rhs.Base)/2;
        int TotalHeight = (Height + rhs.Height)/2;

        return Rectangle(TotalBase, TotalHeight);
    }
}
```

```
//Circle.h
#ifndef CIRCLE_H
#define CIRCLE_H

#include "Shape.h"

namespace Shape {

    class Circle : public Shape{
        float Radius;
    public:
        Circle():Radius(5), Shape(30, 30, Yellow){}
        Circle(float radius, int x, int Y, COLOR color):Radius(radius), Shape(X
            , Y, color){}
        Circle(float radius);
        ~Circle();
        virtual float getArea();
        virtual void drawObject();
        void setSize(float radius);
        float getSize();
        void setColor(COLOR color);
        COLOR getColor();
        void setCenter(int xCoord, int yCoord);
        void getCenter();
        virtual void getFacts();
        Circle operator +(const Circle &rhs);     //add two Circles
        float Area;

    };


}

#endif
```

```
//Circle.cpp
#include "Circle.h"
#include <iostream>
using namespace std;

namespace Shape{
    Circle::Circle(float radius):Radius(radius), Shape(30, 30, Yellow){
        //blank
    }
    Circle::~Circle(){
        //blank
    }
    float Circle::getArea(){
        float PI = 22/7;

        Area = PI * Radius * Radius;
    }
    void Circle::drawObject(){
        cout << "Drawing Circle" << endl;
    }
    void Circle::setSize(float radius){
        Radius = radius;
    }
    float Circle::getSize(){
        return Radius;
    }
    void Circle::setColor(COLOR color){
        col = color;
    }
    COLOR Circle::getColor(){
        return col;
    }
    void Circle::setCenter(int xCoord, int yCoord){
        X = xCoord;
        Y = yCoord;
    }
    void Circle::getCenter(){
        cout << "X = " << X << endl;
        cout << "Y = " << Y << endl;
    }

    void Circle::getFacts(){
        cout << "Coordinates are: " << endl;
        cout << "X = " << X << endl;
        cout << "Y = " << Y << endl;
        cout << "Size: " << endl;
        cout << "Radius = " << Radius << endl;
        cout << "Color: " << col << endl;

    }

    Circle Circle::operator +(const Circle &rhs){
        float PI = 22/7;
        float TotalRad = Radius * rhs.Radius;

        return Circle(TotalRad);
    }
```

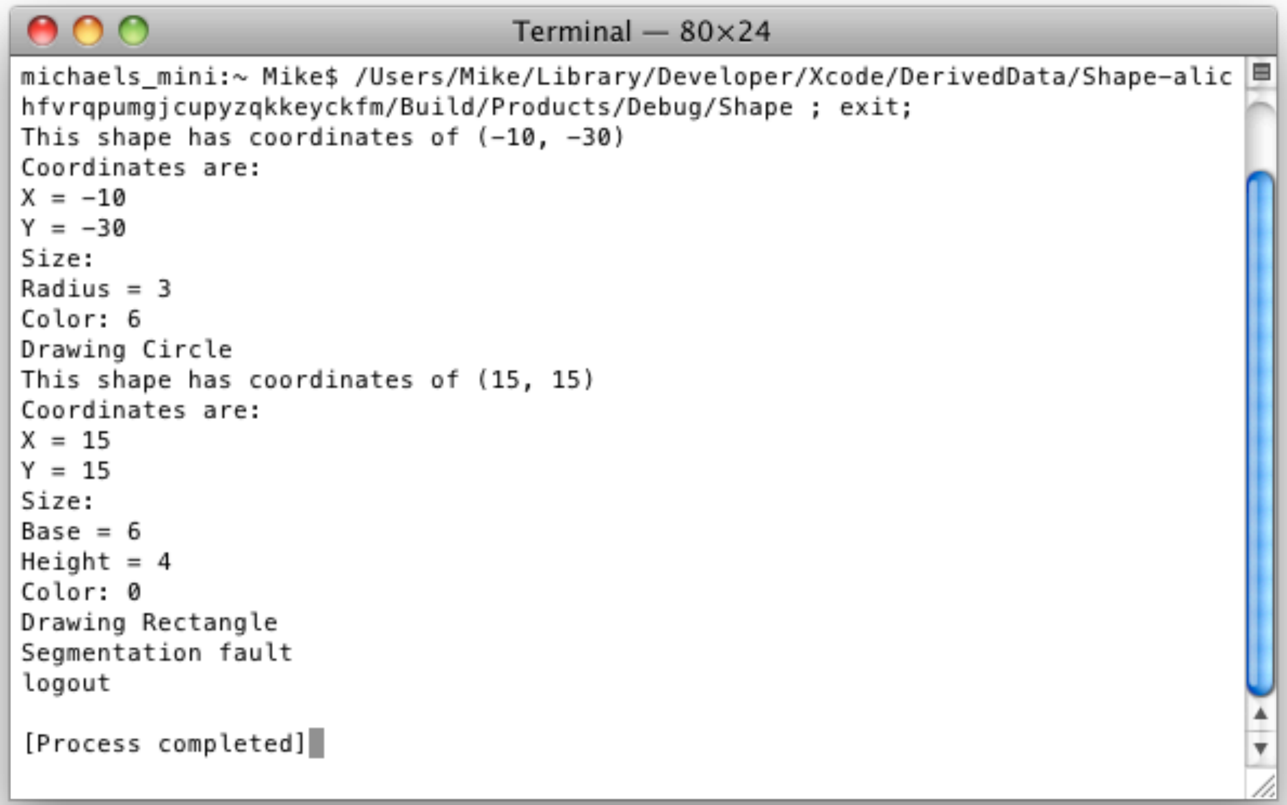
```
//Square.h
#ifndef SQUARE_H
#define SQUARE_H

#include "Shape.h"

namespace Shape {

    class Square : public Shape{
        int Base;
    public:
        Square():Base(7), Shape(0, 0, Green){}
        Square(int base);
        Square(int base, int x, int y, COLOR color):Base(base), Shape(x, y,
            color){}
        ~Square();
        virtual float getArea();
        virtual void drawObject();
        void setSize(int base);
        int getSize();
        void setColor(COLOR color);
        COLOR getColor();
        void setCenter(int xCoord, int yCoord);
        void getCenter();
        virtual void getFacts();
        Square operator +(const Square &rhs);     //add two Squares
        float Area;

    };


}

#endif
```

```
//Square.cpp
#include "Square.h"
#include <iostream>
using namespace std;

namespace Shape {

    Square::Square(int base):Base(base), Shape(0, 0, Green){
        //blank
    }
    Square::~Square(){

    }
    float Square::getArea(){
        Area = Base * Base;

        return Area;
    }
    void Square::drawObject(){
        cout << "Drawing Square" << endl;
    }
    void Square::setSize(int base){
        Base = base;
    }
    int Square::getSize(){
        return Base;
    }
    void Square::setColor(COLOR color){
        col = color;
    }
    COLOR Square::getColor(){
        return col;
    }
    void Square::setCenter(int xCoord, int yCoord){
        X = xCoord;
        Y = yCoord;
    }
    void Square::getCenter(){
        cout << "X = " << X << endl;
        cout << "Y = " << Y << endl;
    }

    void Square::getFacts(){
        cout << "Coordinates are: " << endl;
        cout << "X = " << X << endl;
        cout << "Y = " << Y << endl;
        cout << "Size: " << endl;
        cout << "Base = " << Base << endl;
        cout << "Color: " << col << endl;
    }


    Square Square::operator +(const Square &rhs){     //add two Squares
        int TotalBase = Base * rhs.Base;

        return Square(Base);
    }
```

```
//Triangle.h
#ifndef TRIANGLE_H
#define TRIANGLE_H

#include "Shape.h"

namespace Shape{

    class Triangle : public Shape{
        int Base;
        int Height;
    public:
        Triangle():Base(5), Height(10), Shape(50, 50, Blue){}
        Triangle(int base, int height);
        Triangle(int base, int height, int x, int y, COLOR color):Base(base),
            Height(height), Shape(x, y, color){}
        ~Triangle();
        virtual float getArea();
        virtual void drawObject();
        void setSize(int base, int height);
        void getSize();
        void setColor(COLOR color);
        COLOR getColor();
        void setCenter(int xCoord, int yCoord);
        void getCenter();
        virtual void getFacts();
        Triangle operator +(const Triangle &rhs);     //add two Triangles
        float Area;

    };
}

#endif
```

```
//Triangle.cpp
#include "Triangle.h"
#include <iostream>
using namespace std;

namespace Shape {

    Triangle::Triangle(int base, int height):Base(base), Height(height), Shape
        (50, 50, Blue){
        //blank
    }
    Triangle::~Triangle(){
        //blank
    }
    float Triangle::getArea(){
        Area = (Base * Height)/2;

        return Area;
    }
    void Triangle::drawObject(){
        cout << "Drawing Triangle" << endl;
    }
    void Triangle::setSize(int base, int height){
        Base = base;
        Height = height;
    }
    void Triangle::getSize(){
        cout << "Base = " << Base << endl;
        cout << "Height = " << Height << endl;
    }
    void Triangle::setColor(COLOR color){
        col = color;
    }
    COLOR Triangle::getColor(){
        return col;
    }
    void Triangle::setCenter(int xCoord, int yCoord){
        X = xCoord;
        Y = yCoord;
    }
    void Triangle::getCenter(){
        cout << "X = " << X << endl;
        cout << "Y = " << Y << endl;
    }

    void Triangle::getFacts(){
        cout << "Coordinates are: " << endl;
        cout << "X = " << X << endl;
        cout << "Y = " << Y << endl;
        cout << "Size: " << endl;
        cout << "Base = " << Base << endl;
        cout << "Height = " << Height << endl;
        cout << "Color: " << col << endl;
    }


    Triangle Triangle::operator +(const Triangle &rhs){     //add two Triangles
```

```
        int TotalBase = (Base + rhs.Base)/2;
        int TotalHeight = (Height + rhs.Height)/2;

        return Triangle(TotalBase, TotalHeight);
    }
}
```


Terminal — 80×24

```
michaels_mini:~ Mike$ /Users/Mike/Library/Developer/Xcode/DerivedData/Shape-alic
hfvrqpumgjcupyzqkkeyckfm/Build/Products/Debug/Shape ; exit;
This shape has coordinates of (-10, -30)
Coordinates are:
X = -10
Y = -30
Size:
Radius = 3
Color: 6
Drawing Circle
This shape has coordinates of (15, 15)
Coordinates are:
X = 15
Y = 15
Size:
Base = 6
Height = 4
Color: 0
Drawing Rectangle
Segmentation fault
logout

[Process completed]
```