```
struct Point
{
    float x, y;
};

class Shape
{
public:
    virtual float area() = 0;
    virtual Point & center() = 0;
    std::string name()
    {
        return "Shape";
    }
};

class Circle : public Shape
{
    Point c;
    float r;
public:
    Circle(Point C, float R) :c(C), r(R) { }
    float area()
    {
        return 3.14156265359 * r * r;
    }
    Point & center()
    {
        return c;
    }
    std::string name()
    {
        return "Circle";
    }
};

class Rectangle : public Shape
{
    Point ul, lr;
public:
    Rectangle(Point UL, Point LR):ul(UL),lr(LR)
    {
    }
    float width()
    {
        return lr.x - ul.x;
    }
    float height()
    {
        return lr.y - ul.y;
    }
```

allow derived class to override

not a place to define a method.

class, not type

does not match return type of method

address required as return type

won't match return type

can't see why this is necessary will cause confusion for apparently no gain

```
    float area()
    {
        return width()*height();
    }
    Point & center()
    {
        return Point((lr.x - ul.x) / 2, lr.y - ul.y / 2);
    }
    std::string name()
    {
        return "Rectangle";
    }
};

class Polygon : public Shape
{
    Point * points;
    int n;
public:
    Polygon()
    {}
    Polygon(const Point * p, int N)
    {
        n = N;
        points = new Point[N];
        for (int i = 0; i < N; ++i)
            points[i] = p[i];
    }
    void addPoint(const Point & p)
    {
        points[n++] = p;
    }
    ~Polygon()
    {
        delete points;
    }
    float area()
    {
        float area = 0;
        int j = n - 1;
        for (int i = 0; i < n; j++)
        {
            area += (points[j].x + points[i].x]) * (points[j].y - points[i].y);
            j = i;
        }
        return area / 2;
    }
    Point & center()
    {
        Point c;
        for (int i = 0; i < n; ++i)
```

Can't see good reason for this

No instantiation

Returns local variable but method signature calls for address of Point value in Stack memory

doesn't seem to have a purpose

overloaded function might combined w/ other functions of same name

impossible to discern what this does from looking at it.

it seems like any method can alter "n" however it wishes

need descriptive variable names

would be less confusing to use this -> n to eliminate confusion w/ local variables

that almost always ends in tears. i and j in same for loop

would help to explain why

should use consistent notation

```
        {
            c.x += points[i].x;
            c.y += points[i].y;
        }
        c.x /= n;
        c.y /= n;
        return c;
    }
    std::string name()
    {
        return "Polygon "+n;
    }
};
```

this seems to just over write the same values.

method signature calls for address of this value

this is a class, can't serve as return type

mixed strings and int values. can't.