Name:	
(as it would appear on official course roster)	
Umail address:	@umail.ucsb.edu
Optional: name you wish to be called if different from name above.	
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone")	
section	

1

h02

CS24 F19

h02: Chapter 2.3-2.5

ready?	assigned	due	points
true	Wed 10/02 12:00AM	Wed 10/09 11:59PM	100

You may collaborate on this homework with AT MOST one person, an optional "homework buddy".

INSTEAD OF TURNING IN THIS HOMEWORK, YOU WILL TAKE A QUIZ ON GAUCHOSPACE BY THE DUE DATE.
There is NO MAKEUP for missed homework assignments.
The quiz will be made available at least two days before the due date.

Please:

- **No Staples**.
- **No Paperclips**.
- **No folded down corners**.

Complete your reading of Chapter 2, section 2.3 - 2.5 (If you don't have a copy of the textbook yet, there is one on reserve at the library under "COMP000-STAFF - Permanent Reserve").

1. (30 pts) Consider the point class defined and implemented on pages 64 and 65. Then, answer the following questions:

a. Name all the special class methods that are used to initialize an object of the class at the time that it is created.

point(double initialx, double initialy);

b. What is the output of the following code? Assume the code is embedded in a correct C++ program.

```
point p1, p2(20.0), p3(50,60);
cout<<p1.get_x()<<" "<<p1.get_y()<<endl;
cout<<p2.get_x()<<" "<<p2.get_y()<<endl;
cout<<p3.get_x()<<" "<<p3.get_y()<<endl;
p1 = p3;
p1.rotate90();
cout<<p1.get_x()<<" "<<p1.get_y()<<endl;
```

```
0 0
20 0
50 60
60 -50
```

c. Is the copy constructor called in the above code? If so, where?

no, but the copy assignment operator is called

2. (10 pts) On page 72, why are the parameters of the distance() function of type const reference?

const because the points are not being changed by the function, and reference for efficiency (so that a copy isn't made)

2

h02

CS24 F19

3. (30 pts) For the point class on page 64 and 65, implement the overloaded parameter '*', so that the operator can be used to scale the x and y values of a point by a fixed number. Usage: point p2 = 2.0*p1; In this example p2's x and y coorinates should be double that of p1. Specify whether you would implement the operator as a member function, a non-member function or a friend function, and why?

```
point operator*(double d, const point &p) {
   point ret(d * p.get_x(), d * p.get_y());
   return ret;
}
```

This function can't be a member function of point because the first argument of the * operator must be a double instead of a point. Though this function could have been a friend function, it wasn't necessary because the get_x and get_y functions supplied all the necessary information about the point to create the scaled one.

4. (30 pts) For the point class on page 64 and 65, implement the overloaded parameter '+=', so that for two point objects p1 and p2, p1+=p2, produces the effect of adding p2's x and y member variables to the corresponding member variables of p1, and store the resulting values back into p1. Usage: p1+=p2; Specify whether you would implement the operator as a member function, a non-member function or a friend function, and why?

```
void point::operator+=(const point &other) {
   x += other.x;
   y += other.y;
}
```

This needs to either be a member function or a friend function, because it must update the x and y member variables of the first point (and there is no set_x and set_y function). I chose to implement it as a member function because the operator in my mind "acts on" the first point.