CS 32, Spring 2013
Hw5: 50 total points

Print this form, staple loose pages together, and write your answers on it.

Accepted: program by `turnin` before your lab section on Wednesday, May 15,
plus this paper at the beginning of your lab section on May 15.
No email submission allowed.

Name (1 pt): _____

Umail (1 pt): _____@umail.ucsb.edu

Lab Section (1 pt) Circle one: 1:00 2:00 3:00

Note: it is necessary that you have completed all of the work for Lab05 before beginning this homework, as you will be adding features to the Vec3 class you built in that lab. We also suggest you perform the "After lab-work" tasks of that lab to prepare for this work.

Add the following features to class Vec3 from Lab05. After implementing each one, either place a check mark on the line if you (and your partner) succeeded and incorporated that feature, or write a brief explanation telling us why not. Use the main2.cpp testing program described in the "After lab-work" of Lab05, and check your results against the ones posted at the end of that lab.

1. (9 pts.) Overload += as a member function, to add the components of a constant Vec3 object to the components of the calling object. For example, `Vec(1, 2, 3) += Vec(4, 4, 4)` changes the one on the left to Vec(5, 6, 7).

#1 done: _____
(check, or say why not)

2. (20 pts.) Overload ++ twice, so that it works both as a pre-increment and post-increment operator. For example, both ++vec and vec++ should increase each component of the vec by 1, but ++vec will return that incremented object while vec++ will return a copy of the pre-incremented object. The difference between the signatures of these operators, strangely enough, is that the post-increment version is identified by an int parameter; the other one has no parameters. Also the pre- version returns a reference, but the post- version returns an object.

#2 done: _____

3. (18 pts.) Overload [] twice, so that it works for const and non-constant element access. That is, given a Vec3 object named v, v[0] should return the value of v.x, v[1] returns v.y's value, and v[2] returns v.z's value in both functions. But the constant version should return a copy, whereas the mutable version should return a reference that allows this item to appear on the left side of an assignment:

```
v[0] = 7.5; // uses the mutable version
cout << v[0]; // uses the constant version
```

You may assume that only 0, 1 or 2 will be passed (no need for error-checking).

#3 done: _____

End of Hw5 (but see turn-in instructions on next page)

Turn-in instructions:

A. After testing to verify results are correct, use the turnin program to submit both of your revised vec.h and vec.cpp as follows:

```
turnin hw5@cs32 vec.h vec.cpp
```

If you worked with a partner, then just turn in one copy for both of you, and enter your partner's name on the following line:

_____

B. Each student must turn in this completed paper at your next lab.