CS 60
FOURTH (ACTUALLY FIFTH) QUIZ
May 11, 2010
WRITE ALL YOUR ANSWERS ON SPACE PROVIDED.
ANSWER ALL FOUR QUESTIONS. TOTAL POINTS ARE 30.
YOU MAY ASSUME THAT EACH SECTION OF CODE BELOW IS IN ITS OWN FILE WHEN COMPILED

NAME:_____

1 { Circle for each part True or False depending whether or not the statement is true or false. Each question is worth 1.5 Point}

- { **True** or False } A destructor for a class `y` is invoked when we use `delete xxx` and `xxx` is a pointer to an object from `class y`.
- { **True** or False } Every class has a `constructor`.
- { **True** or False } The copy constructor for `class Player` is invoked when we have `Player p1 = p2;` and `p2` is an object of `class Player`.
- { True or **False** } In `C++` we use `new` to allocate space from the area of memory called the Stack.
- { True or **False** } Overloading the assignment (=) operator for a class is the same as defining a copy constructor for that class.
- { **True** or False } It is possible to overload the equivalence operator (==) for a class to compare two instances of the class.
- { True or **False** } If a member variable is declared as `protected` in class `A` derived from class `B`, then it can be accessed directly (as if it were `public`) in all the member functions of class `B`.
- { True or **False** } In `C++` we use `delete` to delete memory acquired through `calloc` or `malloc`.
- { True or **False** } When a variable is declared as `private` in a class, it can be accessed in all the member functions of that class, except for the member functions that are defined as `inline`.
- { True or **False** } When we define `int& xxx;` we mean that `xxx` is a pointer to an `int`. I.e., to access that value we will use `*xxx`.
- { **True** or False } Every class has one or more `destructors`.

- { **True** or False } It is possible to overload the equivalence operator (==) for a class to compare one instance of a class to an integer.
- { **True** or False } If we have a class called Base, then the line `Base f(b);` invokes the copy constructor for that class if `b` is an object of class Base.
- { True or **False** } If we have a class called Base, then the line `a=b;` results in an invokation of the copy constructor for class Base if both `a` and `b` are objects of class Base.
- { **True** or False } It is possible to define a class member variable as `static`, in which case there will be only one instance of the variable (no matter how many instances of the class we create).
- { True or **False** } In C++ every class needs to have a constructor and destructor defined by the user.
- { **True** or False } The copy constructor for the class `Base` is NOT invoked when we write `Base *g = &a;` and `a` is an object of class `Base`.
- { **True** or False } The equivalence operator (==) for a class is not automatically generated.
- { True or **False** } In `C++` you are not allowed to use `calloc`.
- { **True** or False } In `C++` we use `free` to delete memory acquired through `calloc` or `malloc`.