Grade: 15/20

**Homework 3**

**Focus**

* Classes
* Association

**Problem:**

We will model a game of medieval times. Our world is filled with nobles and warriors. Nobles don't have much to do except do battle with each other. (We'll leave the feasting and other entertainments for add-ons.) Warriors don't have much to do except hire out to a noble and fight in his behalf. Of course the nobles are pretty wimpy themselves and will lose if they don't have warriors to defend them. How does all this work?

* Warriors start out with a specified strength.
* A battle between nobles is won by the noble who commands the stronger army.
* The army's strength is simply the combined strengths of all its warriors.
* A battle is to the death. The losing noble is dead as are his warriors.
* The winner does not usually walk away unscarred. All his men lose a portion of their strength equal to the ratio of the enemy army's combined strenth to their army's. If the losing army had a combined strength that was 1/4 the size of the winning army's, then each soldier in the winning army will have their own strength reduced by 1/4.

A test program and output are shown below.

**Programming Constraints**

What would a homework assignment be without unnecessary and unreasonable constraints?

Your classes must satisfy the following:

* The battle method will announce who is battling whom, and the result.
  + If one or both of the nobles is already dead, just report that.   
    The "winner" doesn't win anything for kicking around a dead guy. And his warriors don't use up any strength.
  + Look at the output for the sample test program to see what you should be displaying.
* A noble's army is a **vector of pointers** to warriors.

**And some things to make life easier**

Sure it's more fun to make you do lots of work, but I decided to give you a break on error checking. You may assume that:

* If a noble hires a warrior, both already exist and the warrior has not been hired by anyone else.
* If two nobles do battle, they both already exist (though no guarantees that they are still alive).

**Turn in**

Hand in the file named hw03.cpp.

**Sample Test Code**

int main() {

Noble art("King Arthur");

Noble lance("Lancelot du Lac");

Noble jim("Jim");

Noble linus("Linus Torvalds");

Noble billie("Bill Gates");

Warrior cheetah("Tarzan", 10);

Warrior wizard("Merlin", 15);

Warrior theGovernator("Conan", 12);

Warrior nimoy("Spock", 15);

Warrior lawless("Xena", 20);

Warrior mrGreen("Hulk", 8);

Warrior dylan("Hercules", 3);

jim.hire(nimoy);

lance.hire(theGovernator);

art.hire(wizard);

lance.hire(dylan);

linus.hire(lawless);

billie.hire(mrGreen);

art.hire(cheetah);

jim.display();

lance.display();

art.display();

linus.display();

billie.display();

art.fire(cheetah);

art.display();

art.battle(lance);

jim.battle(lance);

linus.battle(billie);

billie.battle(lance);

}

Output for Sample Code

Jim has an army of 1

Spock: 15

Lancelot du Lac has an army of 2

Conan: 12

Hercules: 3

King Arthur has an army of 2

Merlin: 15

Tarzan: 10

Linus Torvalds has an army of 1

Xena: 20

Bill Gates has an army of 1

Hulk: 8

Tarzan, you're fired! -- King Arthur

King Arthur has an army of 1

Merlin: 15

King Arthur battles Lancelot du Lac

Mutual Annihalation: King Arthur and Lancelot du Lac die at each other's hands

Jim battles Lancelot du Lac

He's dead Jim

Linus Torvalds battles Bill Gates

Linus Torvalds defeats Bill Gates

Bill Gates battles Lancelot du Lac

Oh, NO! They're both dead! Yuck!

Grader comments:

Do not have a public method that returns a noble's army. This breaks encapsulation.  
vector<Warrior\*> getWarriors()  
    {  
        return warriors;  
    }

In the hire method, the pointers should really point to the object references passed to this method. You **should not**create a copy of the warrior on the heap and store the pointer to that object in the army.

void hire(Warrior& combatant)  
    {  
        Warrior\* employee = new Warrior(combatant.getName(), combatant.getStr()); this is wrong. It should be warriors.push\_back(combatant)  
        warriors.push\_back(employee); //add warrior to vector  
    }

Noble should also have a field isAlive to track the alive status of a noble.

In void battle(Noble& otherNoble),

ratio should be of type double. Else, this will always evaluate to 0.

else if (warriors.size() == 0 && otherNoble.getWarriors().size() == 0)  
        {  
            cout << "Oh, NO! They're both dead! Yuck!" << endl;  
        }  
        else if (warriors.size() == 0 && otherNoble.getWarriors().size()>0)  
        {  
            cout << "He's dead" << otherNoble.getName() << endl;  
        }  
        else if (warriors.size()> 0 && otherNoble.getWarriors().size() == 0)  
        {  
            cout << "He's dead" << name << endl;  
        }

These conditions are wrong. You should, check the strength of the noble and not the size of the army as you are not removing the warriors from the army once they are dead.

In the fire method, the loop should stop once the warrior is found and fired. Use break to accomplish this.

for (size\_t i = 0; i < warriors.size(); i++)  
        {  
            if (combatant.getName() == warriors[i]->getName())  
            {  
                warriorIdx = i;  
            }  
        }

Make display and getters const methods.