Assignment #3 – Classes/Inheritance
Due: Sunday, 05/14/17, 11:59pm

Grading: For each programming assignment, you are graded by explaining and demoing your code to a TA. **You must demo your program BEFORE the next assignment is due**, and if you fail to do so, you will automatically lose 50 points! **Your job is to convince the TA that your program works correctly, i.e. show your TA how to use/break your program☺**

(90 pts) **Problem Statement:** This is a classic inheritance assignment but with a twist!!! Suppose that you are creating a fantasy role-playing game. In this game, we have four different types of creatures: humans, cyberdemons, balrogs, and elves. To represent one of these creatures, one might define a Creature class as follows:

```
class Creature {
   private:
      int type; // 0 human, 1 cyberdemon, 2 balrog, 3 elf
      int strength; // How much damage we can inflict
      int hitpoints; // How much damage we can sustain
      string name;  //The name of the creature
      double payoff;  //How much you win for a creature
      double cost;  //How much it cost to purchase the creature
      string getSpecies(); // Returns type of species
   public:
      Creature( );
      // Initialize to human, 10 strength, 10 hit points

      Creature( int newType, int newStrength, int newHit);
      // Initialize creature to new type, strength, hit points

      // Also add appropriate accessor and mutator functions
      // for type, strength, hit points, etc.

      int getDamage();
      // Returns amount of damage this creature
      // inflicts in one round of combat
};
```

This would be the corresponding implementation of the `getSpecies`( ) function:

```
string Creature:: getSpecies() {
   switch ( type) {
      case 0: return " Human";
      case 1: return " Cyberdemon";
      case 2: return " Balrog";
      case 3: return " Elf";
   }
   return " Unknown";
}
```

The `getDamage( )` function outputs and returns the damage this creature can inflict in one round of combat. The rules for calculating the damage are as follows:

- Every creature inflicts damage that is a random number r, where `0 < r <= strength`.
- Demons have a 5% chance of inflicting a demonic attack, which is an additional 50 damage points. Balrogs and Cyberdemons are demons.
- Elves have a 10% chance to inflict a magical attack that doubles the normal amount of damage.
- Balrogs are very fast, so they get to attack twice.

This would be the corresponding implementation of `getDamage`( ) for this design:

```
int Creature::getDamage( ) {
   int damage;

   // All creatures inflict damage, which is a
   // random number up to their strength
   damage = ( rand( ) % strength) + 1;
   cout << getSpecies( ) << " attacks for "
        << damage << " points!" << endl;

   // Demons can inflict damage of 50 with a 5% chance
   if (( type = 2) || ( type == 1))
      if (( rand( ) % 100) < 5) {
         damage = damage + 50;
         cout << " Demonic attack inflicts 50 "
           << " additional damage points!" << endl;
      }

   // Elves inflict double magical damage with a 10% chance
   if ( type == 3) {
      if (( rand( ) % 10)== 0) {
         cout << " Magical attack inflicts " << damage
              << " additional damage points!" << endl;
         damage = damage * 2;
      }
   }

   // Balrogs are so fast they get to attack twice
   if ( type == 2) {
      int damage2 = ( rand() % strength) + 1;
      cout << " Balrog speed attack inflicts " << damage2
           << "    additional damage points!" << endl;
      damage = damage + damage2;
   }
   return damage;
}
```

One problem with this implementation is that it is unwieldy to add new creatures!

Rewrite the class to use inheritance, which will eliminate the need for the variable type. The `Creature` class should be the base class. The classes `Demon`, `Elf`, and `Human` should be derived from `Creature`. The classes `Cyberdemon` and `Balrog` should be derived from `Demon`. You will need to rewrite the `getSpecies( )` and `getDamage( )` functions so they are appropriate for each class.

For example, the `getDamage`( ) function in each class should only compute the damage appropriate for that object. The total damage is then calculated by combining the results of `getDamage`( ) at each level of the inheritance hierarchy. As an example, invoking `getDamage`( ) for a `Balrog` object should invoke `getDamage`( ) for the `Demon` object, which should invoke `getDamage`( ) for the Creature object. This will compute the basic damage that all creatures inflict, followed by the random 5% damage that demons inflict, followed by the double damage that balrogs inflict.

We will have a `World` class that has a **dynamic array of each type of creature**, Human, Elf, Cyberdemon, and Balrog, and **how much money you have in your world**. The user can add money and creatures to their world.

Write a driver that will create a World and determine from the user how much money they have in their world to start, as well as the number of each type of creature they want to buy and insert into their world. The user must provide a name for each creature they enter into their world, and the number of each creature type inserted into the world will determine how much money is left in the world. Now, the battle begins!!!

In each round, decide which creatures are going to battle, and the damage inflicted by each creature is subtracted from the other creature's hit points. Creatures with hit points > 0 can continue to battle in subsequent rounds, but creatures with hit points <= 0 are removed from the world. The last creature with hit points > 0 is the winner, and the payoff is added to the bank. You need to continue to play rounds until a winner is determined, but the user has the option to buy new creatures to add to the world at the end of each round.

After a winner is determined, you need to print the winner's name and creature type, as well as the amount of money in the bank, which may be positive (the games owes you) or negative (you owe the game)!!!

You can determine how you want to setup the Creatures strength, hit points, payoff, and cost. You may want to make these a constant amount depending on the creature type, you may want to set them to a specific amount depending on the user's choice or a random number (or any combination of these)! These are decisions that will make your game unique! For example, you may allow different humans to have different strengths with different hit points (how much they can endure) and the cost and payoff might be different based on these values. Be creative! ☺

Requirements:

- You are required to have the needed constructors, appropriate use of const, and the appropriate use of the Big Three.
- All member variables need to be private and/or protected, and you must have accessor and mutator functions for each member in the class.
- Appropriate behavior for each object to correctly implement the creature battle functionality.
- You must have a Makefile, .h and .cpp for each class, and a driver file with the main function.
- Your program must not have a memory leak!!!
- Keep your functions under 20 lines!!!

Extra Credit: 10 pts

Add a new creature to your program with a new damage calculation.

(10 pts) Program Style/Comments

In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Below is an example header to include. Make sure you review the style guidelines for this class, and begin trying to follow them, i.e. don't align everything on the left or put everything on one line! http://classes.engr.oregonstate.edu/eecs/spring2017/cs162-001/162_style_guideline.pdf

```
/*********************************************************
** Program: Creature.cpp
** Author: Your Name
** Date: 05/02/2017
** Description:
** Input:
** Output:
*********************************************************/
```

Electronically submit your C++ program (**.h and .cpp files and Makefile,** not your executable!!!) and **as a tarred archive** by the assignment due date, using TEACH.