Assignment 2 – Inheritance & Polymorphism
CS 360

Date Assigned: Wednesday, April 18th, 2018
Date Due: Wednesday, May 2nd, 2018 @ 1:00pm
Submit via Canvas and turn in a printout

Points Possible: 100

Assignment Description
In this assignment, you will write a program that simulates a group of Orcs, Trolls, and Goblins wandering the countryside and terrorizing random villagers.

You will create an abstract class Enemy as the super class and Goblin, Troll, and Orc as sub classes that inherit from Enemy.

Variables
Each enemy has the following variables:
- A unique integer id (this value is unique across all types of enemies)
- Current health (int)
- Strength (int)
- Constitution (int)
- Starting x location (int)
- Starting y location (int)
- A boolean indicating if this enemy is alive or dead (true = alive, false = dead)

You may include any other member variables you feel are necessary.

Functions
Each enemy has the following operations:
- isAlive – returns the bool indicating if the enemy is alive or dead
- *update – updates the enemy's location. Each type of enemy moves differently, see below.
- *attack – this function represents the enemy attacking some random passerby. Each type of enemy attacks differently, see below.
- *injure – the random passerby fights back! See below for details.
- *print – displays information about the enemy. See below for details.
- the constructor – see below for details

Functions marked with an '*' are pure virtual in Enemy.

You may add other additional operations you feel are necessary. Mark appropriate functions and variables as `const` or `static`.

NOTE: If a function is both **`pure virtual`** and **`const`**, the **`const`** keyword must appear before the '**`=0`**' pure specifier. For example,

```
virtual float calculateArea() const = 0;          //correct
```

not

```
virtual float calculateArea() = 0 const;          //syntax error
```

Function Details

Constructor
The constructor receives the enemy's starting health, strength, constitution, starting x location, and starting y location. This information comes from a file enemyInfo.txt. See below for details.

The constructor assigns an unique id value to the enemy, assigns all the member variables based on the parameters, and sets the enemy to be alive. Finally, the print function is called.

NOTE: Don't forget that you can have the child class call the super class constructor to handle some operations and have the sub class constructor do others.

Update
This function changes the variables associated with the enemy according to the following rules:

- Goblins move in the x direction between -3 and 3 units and in the y direction between -2 and 2 units per turn
- Trolls don't like to move north or south. They move only in the x direction for between 7 and 10 or -7 and -10 units. They also regenerate health equal to their constitution score each time that update is called. The troll's health should not exceed their max health that comes from the file.
  - Print out a message indicating how much health was recovered and the troll's new health.
- Orcs like to move diagonally. They pick a value between -5 and 5 units and move the same amount in both the x and y directions.

Attack
The enemy attacks a random passerby according to the following rules. The notation 1d# indicates a die with # sides. It produces random values between 1 and #. For example, 1d4 means a random value between 1 & 4 (inclusive).

- Goblins do damage equal to 1d4 + strength.
- Trolls do damage equal to 1d8 + strength
- Orcs do damage equal to 1d6 + strength

For example, if an Orc has a strength of 5, it can do anywhere between 6 and 11 damage on an attack.

After calculating damage, print out a message that includes the type of enemy, its id value, and the amount of damage it inflicted. For example,

```
Orc 3 attacks a random passerby for 10 damage!
```

Injure
The random passerby doesn't take too kindly to being attacked. This function receives a random value between 0 and 10 that represents the damage the passerby inflicts on the enemy. However, each enemy can reduce the amount of damage they actually take based on their constitution score.

- Goblins are squishy. Subtract half their constitution from the damage.

- Trolls are much harder to defeat. Multiply their constitution by 1.5 and subtract it from the damage.
- Orcs are of average toughness. Simply subtract their constitution from the damage.

After performing this calculation, it's possible that the enemy managed to reduce the damage taken to a value <= 0. If so, do not change their health. Print out a message indicating that the attack didn't work. This message must include the type and id. For example,

```
The passerby tries to attack Troll 1, but it's not very effective...
```

If the enemy does take damage, subtract that from their health. Print out a message that includes the type of enemy, its id value, the damage taken, and its new health. For example,

```
Troll 4 takes 5 damage! Current hp = 17
```

If the enemy's current health is <= 0, print an additional message that the enemy has been defeated.

```
Goblin 2 has been slain!
```

Print
This function prints a message in the following format. **No other formats are allowed.**

```
type id @ (xLocation, yLocation) hp = currentHealth
```

For example,

```
Goblin 7 @ (14, 8) hp = 10
```

Main

This file has been provided for you. You may make changes to main.cpp while testing, but the final program you turn in MUST compile and run with the provided main.cpp. Look this over before you start. It will give you some ideas about how your classes should be coded.

The Input File

The program requires a file named “enemyInfo.txt”. The first line of the file indicates the number of enemies present in the file. The other lines contain the following values about a single enemy, separated by spaces:

- A value indicating the type. 0=Goblin, 1=Troll, 2=Orc
- The enemy's starting health
- The enemy's strength
- The enemy's constitution
- The enemy's starting x location
- The enemy's starting y location

For example,

```
0 20 3 2 5 2
```

is a Goblin with 20 health, 3 strength, 2 constitution, and starts at (5, 2). A sample enemyInfo.txt has been provided for you.

Other Requirements, Hints, and Tips

1. Your submission should include eight files: a .h and .cpp for **Enemy, Goblin, Orc, and Troll**.
   a. You do not need to turn in main.cpp.
2. Programs that don't compile with the provided main.cpp will not receive any credit.
3. You can assume that the input file is in the correct format.
4. Aside from the print function, you can format the output however you like.
5. No destructors are required for this program.
6. The child classes should inherit as much as possible from Enemy.
7. Start by only implementing Enemy and Goblin. Once that works, add in Troll and Orc one at a time.
8. It's possible that all enemies will die before all the turns are up and nothing will be printed for the remaining turns. That's okay.
9. You can use the **`rand()`** function to generate a random number. The following example generates a random number between 7 and 10. Make sure to include <cstdlib> to use **`rand()`**.

```
int myRandom = rand() % 3 + 7;//random number 7-10
```

10. Test your program on several input files!

Formatting, Comments, and Submission Guidelines

See the document “Programming Assignment Guidelines” on Canvas. Failure to follow the guidelines will result in a loss of credit as detailed in said document.

Sample Output

See the file sampleOutput.txt. The sample enemyInfo.txt was used to generate this output.