Assignment 3 – Polymorphism via Pointers
CS 360

Date Assigned: Monday, May 14th, 2018
Date Due: Monday, May 28th, 2018 @ 1:00pm
Submit via Canvas and turn in a printout

Assignment Description

In this assignment, you will add a GameBoard class to the program you built in Assignment #2. The GameBoard class will display a simple text based Cartesian coordinate system, and indicate where the Enemies are on each turn by placing a letter on the game board.

Furthermore, I will provide you with new Enemy.h, Troll.h, Goblin.h, and Orc.h files. You can and should use your implementations (.cpps) of these classes from Assignment 2 as a starting point. However, note that you will have to change those implementations to match the provided .h files as needed.

Variables

A GameBoard has the following variables:

- x coordinate
- y coordinate
- a collection of Enemies

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

Functions

A GameBoard has the following operations:

- the constructor – see below for details
- addGamePiece()
- printBoard()
- resetBoard()

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

GameBoard Function Details

Constructor
The constructor receives the size of the board to print by accepting the maximum x and y axis values.

addGamePiece()
This function accepts a pointer to an Enemy, and then adds that pointer to the collection of Enemy pointers. This function will be called by main.cpp.

resetBoard()

This function clears out the collection of Enemies. This function will be called by main.cpp at the conclusion of every turn, after the GameBoard has been printed.

printBoard()

This function prints a GameBoard in the following format, using the collection of Enemies. This example shows a simple 5x5 gameboard, and has an uppercase “G”, “T”, or “O” to indicate that an Enemy is in this location and what type the Enemy is. Your coordinate system will be much bigger of course! This function will be called at the conclusion of every turn by main.cpp.

```
.....|.....
.....|.....
.....|..T..
.....|.....
.....|.....
G_____|_____
.....|.....
.....|..O..
.....|.....
.....|.....
.....|.....
```

Main

This file has been provided for you. Note that the main.cpp file for Assignment #3 is slightly different from the one in Assignment #2. Make sure you get the new one. 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 and the provided .h files. Look main.cpp over before you start. It will give you some ideas about how your classes should be coded.

Note that this program makes use of all of the classes and text files that were part of Assignment #2. However, before you start, make a copy of your existing Assignment 2 files.

Changes to Enemy, Orc, Goblin, and Troll.

There are a few changes required to print the “T”, “O”, or “G” on the GameBoard. Specifically, you'll note that Enemy has a pure virtual method called getDisplayChar() that you'll need to provide for each type of Enemy. The GameBoard should call this getDisplayChar() function when it is printing out the GameBoard.

Other Requirements, Hints, and Tips

1. Your submission should include only the following files: **GameBoard.cpp, GameBoard.h, Enemy.cpp, Goblin.cpp, Orc.cpp, and Troll.cpp** and **makefile.**
   a. You do not need to turn in main.cpp or the provided .h files.

2. Consider using the "vector" class for your collection of Enemies. You can find information on it at this link: http://www.yolinux.com/TUTORIALS/LinuxTutorialC++STL.html#VECTOR

3. You Gameboard class should work with GameBoard sizes between 60,60 and 200,200

4. When a monstor dies, you should change his display on the gameboard from uppercase to lowercase. For example, a goblin that is alive would show a 'G', whereas a deceased goblin would show a 'g'.

5. You must not allow your characters to step off the grid. If your character is supposed to move past the edge of the grid, use the edge as the max place they can go. For example if the grid size is 60x60, but your character is supposed to move from 58 to 62, you should have them move from 58 to 60 instead.

6. If two game pieces try to occupy the same x-y location, you must randomly choose a different location that is within 2 locations on the x or y axis. For example, if an orc tries to occupy location 45,50, but there is already a goblin there, then the orc could move to 46,50.

7. Each time your game piece moves, you must display his old and new coordinates in the program output.

8. You must be able to generate a unique id for a particular enemy, and that id must be uniq across all instances of all enemies. For example, if you created 2 goblins, an orc, and 2 trolls, each ID must be unique. So you might have ids 1,2,3,4 and 5.

9. Programs that don't compile with the provided main.cpp and .h files will not receive any credit.

10. You will be required to submit a working makefile. Make sure you include this with your submission.

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 for what the sample output for this program should look like. Note that your output will look different, especially considering that the sample output doesn't deal with requirements 4, 5, 6 and 7.