Assignment 4 – Containers

Goals-

Use linked structures to hold and manipulate data

Modify an existing parent abstract class (for use by the derived classes)

You will make a program to run a tournament. Start with your assignment 3 program for creatures if possible. If you had issues with assignment 3 and need to begin fresh, reference the creature descriptions from assignment 3 to build your fighters. A user will enter the number of fighters both players will use. The first player supplies a lineup of creatures in order. The second player then supplies their lineup of creatures in order. After each round you will display which type of creatures fought and which won. At the end of the tournament your program will display: the first, second, and third place finishers, their teams, as well as which side scored more points. You will provide an option for the use to see a list of the final order of all fighters (this means you will display the loser pile after displaying the first three finishers).

You should be using polymorphism and all creatures should inherit from the Creature class. The users should enter the type of creature and name. Each object should be instantiated and put into their lineup. You should only use pointers to creatures in your program. A creature pointer can then be used to indicate an object of a subclass and polymorphism will ensure the correct function is called. As your TA if you are uncertain what this means.

By lineup I mean something like a batting order in baseball or softball. The head of each lineup fights. The winner gets put at the back of his team's lineup; the other goes to the container for those who lost. The lineup order cannot be changed once the players are entered. Similarly, the loser pile should have the first loser at the top and the last loser at the bottom.

For this program you must use the structures from Lab 6 to hold the lineups and the loser pile. Which (stack-like or queue-like) will you use for the lineup? Which will you use for the loser pile?

Even if a creature wins she may have taken damage. You should restore some percentage of the damage when they get back in line. Make this simple as you are adding a new member function to the parent class. Some examples are: generate a random integer in some range, or on a role of 5 they recover 50% of the damage they lost, or some scheme of your own. If you want to use a different recovery function for each character that is fine. Maybe give the goblins a chance by restoring all their strength points. Remember to use polymorphism!

At the end of the tournament you should display the first, second, and third place finishers. You should also indicate their team(s), and which team won the tournament. This is another element you must define in the analysis and design stage. You must have some method for determining the total points for each team. You do not need to base it only on wins and loses. If a weaker creature beats a stronger creature maybe they get more points? ☺ In your final design document clearly state the system you are using and how you will implement it.

This is a programming assignment and not a complete game! Other than entering the lineup the players just wait for the results. They should take no further actions. Make sure you test your program with instances of all creature types you've created. You may not be able to do an exhaustive test but make it as extensive as you can.

To help your testing and that of the grader provide an option to display the winner and the updated scores for each round. Also provide an option at the end of the tournament to display the contents of the loser pile, i.e. print them out in order with the loser of the first round displayed last.

What to submit in your zip file:

Your files that implement your class hierarchy

Your file(s) that implement the tournament, including the main function

Your makefile

Your reflections, which can include your design document, which as usual discusses the design, assumptions you needed to make, the test plan, testing results, and changes necessary as a result.

NOTE: Testing will be more involved. The reflections has more weight so you can provide a more detailed description of how you tested your program. Remember to include the results of unit testing!

Grading:

- programming style and documentation (10%)
- create tournament, getting the fighters, resolving the round, put the 2 fighters into the appropriate containers (25%)
- prompt the user and fill the two lineups based on the number of fighters entered by the user (10%)
- create and properly use the structures to hold the lineups (10%)
- create and properly use the structures to hold the loser pile (10%)
- create a heal or recovery function for the winners of each round (5%)
- create and implement a system to determine the final three finishers as well as determine which side won the tournament (10%)
- reflections document to include the design description, test plan, test results, and comments about how you resolved problems during the assignment (20%)