Project 4 – Containers

Implement Fantasy Combat Game Tournament

Goals

- Use linked structures to hold and manipulate data
- Modify an existing parent abstract class

Project

In this project, you will write a program to run a tournament. Start with your project 3 program for creatures. If you had issues with project 3 and need to begin fresh, reference the creature descriptions from project 3 to build your fighters.

This is a **one-user-two-players** game. A user will enter the number of fighters both players will use. The user should also enter the type of creatures and fighter names. The first player supplies a lineup of creatures in order. The second player then supplies his lineup of creatures in order. By lineup I mean something like a batting order in baseball or softball. The **head of each lineup fight** in the same way they fight in project 3. The **winner gets put at the back of her/his team's lineup**; the **loser 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 last loser at the top and the first loser at the bottom.

Even if a creature wins, she/he 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 ranges, e.g., on a roll 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. Remember to use polymorphism if the recovery is different for different creatures!

After each round, you will **display which type of creatures fought and which won on screen** (For example, `Round 1: Team A Blue man No.1 VS. Team B Harry Potter No.1, Harry Potter No.1 won!`). At the end of the tournament (when one team or both run out of fighters in the lineup), your program will **display final total points for each team** (you can determine how to score each round of fight, for example, winner team +2, loser team -1, tie +1). You must **display which team won the tournament**. This is another element you must define in the analysis and design stage. You must **have a method to determine the winner** and

you can determine it by yourself. To help you testing and that of the grader's, 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.

You should be using polymorphism and all creatures will inherit from the Creature class. Each object should be instantiated and put into their lineup. You will 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. Ask your TA if you are uncertain what this means.

NOTE: Depending upon your implementation in Project 3, you could have ties. How does that tie change your end of match logic? What do you do if the final match is a tie? Specifically, where do you put the fighters? For this program, you should use stack-like or queue-like structures to hold the lineups and the loser pile. Which will you use for the lineup? Which will you use for the loser pile?

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.

What you need to submit:

- Your files that implement your class hierarchy
- Your files that implement the tournament, including the main function
- Your makefile
- Your reflections, which can include your design, assumptions you needed to make, the test plan, testing results, and changes necessary as a result.
- Pack them all in a zip file and submit it to TEACH.

NOTE: Testing will be more involved in this assignment. The reflection 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, get the fighters, resolve the round, and put the fighters into the appropriate containers: 25%

• Prompt the user to enter the number and type of fighters and fill the two lineups based on the number and type entered by the user: 10%

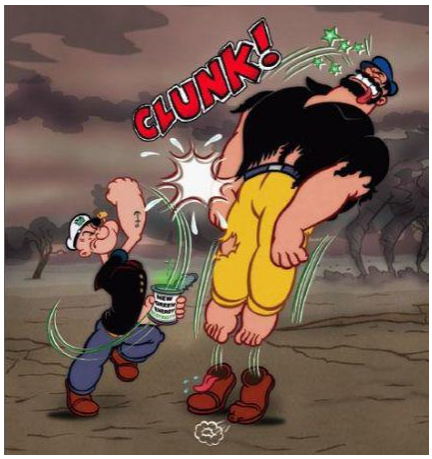
• Create and properly use the structures to hold the lineups: 10%

• Create and properly use the structures to hold the loser pile: 10%

• Create a recovery function for the winners of each round: 10%

• Implement a system to determine which side won the tournament and display the results: 10%

• Reflections document to include the design description, test plan, test results, and comments about how you resolved problems during the assignment: 15%



Popeye adding to the loser pile.