

Assignment 4 – Containers

Goals-

Use appropriate data structure containers from the STL

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 fighters if possible. 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. At the end of the tournament your program will display: the first, second, and third place finishers as well as which side scored more points.

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 may only statically allocate Creature pointers or structures containing creature pointers for your lineup. Ask you TA if you're unclear on static vs dynamic memory allocation.

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.

For this program you must use STL containers to hold the lineups and the loser pile. You may use vectors. If you do make sure the order of the lineups only changes when a fighter is taken out for a bout and the winner is put back at the other end. Similarly, the loser pile should have the first loser at one end and the last loser at the other end. If you want to do a little research you can use the STL queue (first in first out) for the lineups and a stack (last in first out) for the loser pile. The advantage to using a stack and queue is that the management is done for you. That means you have less code to write. ☺

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 roll of 5 they recover 50% of the damage they lost, or some scheme of your own.

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. I recommend doing some research on different standings systems. You must have some method for determining the total points for each team. You do not need to base it only on wins and losses. 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.

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 containers to hold the lineups (10%)
- create and properly use the container 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%)