Design

To create the program, I'm just gonna use the stacks and queues from lab 6, almost exactly for the teams and loser list. Since my functions from the last assignment worked, I can just use those almost exactly as they are, just adding the team and name variable. For the main program, I can just use most the code from the last assignments driver program. I just need to do more to create the lists and loop through for each pair. Then I will need to sort them for printing at the end. I need to keep track of wins as well. But that wont be too difficult, just two integers.

Reflection:

This assignment wasn't too difficult, Since my creatures from the last assignment worked perfectly, It was really just implementing that, and the stuff from lab 6. I did have to modify the creatures in a few ways. For one I had to add a team assignment, as well as a unique name string for each monster. I was able to base my tournament off of the driver program from the last assignment, using the same creature creator, with the addition of adding a name, and using the same battle functions. I did have to change some stuff, like variables passed to different functions. For the regenerating HP, I just had it regenerate a certain amount that was no more then their maximum HP minus their current HP.

In addition to the basic driver function, I had to add stuff for creating teams. This was basically an implementation of what was done in lab 6. I used a queue for the two teams, as I wanted the first creature added to be the first out, and a stack for the loser pile because the top of the stack will be the second place. I was able to also use the same structures for both the stack and the queues, which was nice. I just had to use different add and remove functions, which were just slightly modified from lab 6. The only change from lab 6 was instead of having a char data, I had a Creature pointer. The res of the functions needed for the linked lists I was able to use almost verbatim from lab 6.

Luckily I had very few errors when building the program, the only one really was when including my utilities function. Another one was when adding the regenerating HP, I forgot to make it a member of the creature class so I was getting errors from that. I luckily didn't get any errors form the linked list, which I was surprised by because of all the pointers being passed throughout the lists and the lists themselves. No seg faults at all either.

Error Checking & Testing

My plan for testing will be to check all the inputs by inputing the correct, as well as incorrect inputs. Next I will run through the program multiple times with different creatures, and different size

teams to make sure that it all works. I will go through the program turn for turn to make sure it goes through the turns correct.

For error checking, most of the inputs are characters so using else if statements to check the inputs is what I did. If it wasn't one of the proper inputs, it just loops back through and asks again. The only other input is one for an integer, So I used my utility file which has a function that prompts for a good positive integer and returns it. For testing, I tested all the inputs with the correct ones, as well as bad ones to see what happens. I then added a few cout's to look at the health at every turn. I then ran the program using some different creatures and went through the program turn for turn to make sure it was running correct, and that the displayed winners were correct.

From testing, the program looks to run correctly. The inputs work for all the loops, And the integer input also works. Going through the tournament with a team of two, team two wins and the correct winners are displayed in order. Testing with teams of 3, the team 1 had more wins. I went through it on paper and it is correct.