CS162-400

Assignment 4- Polymorphism + Linked Lists Tournament

- I decided to retest my creature matchup graph, to see if the point system was indeed accurate. It is easy to test in this tournament mode, just set up 10 creates of each type on each team and let the go at it. Most matchup's and scores were representative of the newly tested outcomes; however, it seems as if I had a bit of a fluke run with the following:

  - Reptile vs. Shadow: Out of 20 battles ….. Shadow 1: Reptile 19 (Adjust points to 60/10)
  - Goblin vs. Shadow: Out of 20 battles ….. Shadow 20: Reptile 0 (Adjust points to 60/10)

All other matches were correctly scored.

- I must say that many of the validation functions remained the same, and they work as they should. I tried to make any repeating code into a function as to make everything more smooth. The function howMany only allows users to enter numbers 1-15, any other valued is screed off. The name that users call their creatures, in the function choose can be anything as far as I am concerned, so numbers, spaces, letters, and symbols are all allowed. The user is also entering a 1-5 in the function choose and works similar to the validation function in howMany. It only lets users enter 1-5. no spaces, numbers, symbols.

- Many functions that may require testing, are tested in real time each program is run. I created recursive functions in both the stack and queue classes to display a current list of creatures after each match is over. The creatures are always placed into the correct position. The loser pile is treated as the stack, so the last one in will always appear at the top. The active team lineups work like a queue, and the winner is shown to be added to the end after a victory.

- The selectionSort function always works as intended in real time as well. The program displays the ranking of all creature's points from high to low. As I mentioned in my design, I did this because many times creatures will have much similar scores across the board, so displaying only three would not really be accurate. You can still consider the first three ranked to be the top three, but there are many instance where there are ties.

- The restore function does the trick as well. It will applies 20% of creature's strength when it is down that much. It uses integer variables so any fraction is rounded down. I hard coded a safety guard so that there is no way that the strength can go over the initial maximum value.

- The goblin special is reset at the end of every match in the function getDeath. In the previous version (assignment 3), the goblin static variable (static int goblin) in the creature class would get set back to 1 each time the program would run (because a new creature class would be created, and set the variable to 1 in the constructor). Now, with ongoing matches, I had to create a function to allow the use of the goblin variable to reset after each match. I conducted tests where I hard coded 12 damage into the goblin's attack, and demonstrated that the special would be gone in next matchup.

Thanks for playing!