Elizabeth Poore
Assignment 4 Reflection

Game Rules / Design:

- Assigning battle points:
  - A winning character receives points equal to the strength of the opponent they defeat upon going into battle
  - The points won by a character are added to the overall player tally of points for the tournament
  - Any players surviving at the end of combat are awarded points equal to the remaining strength points of each surviving character
- Restore health functionality:
  - If a character wins the battle but loses strength points in the process, a coin is flipped (bool). If restore is deemed true, the character gets back half of the strength lost in the battle, otherwise their number of strength points beyond the battle remains the same
- Following combat, any characters still living are added to the loser pile so that determination of the top placers can be determined.
  - Determination of top finishers is determined by maximum points scored while in play

Initial Design:

- The initial design for updating my program using linked lists didn't require huge amounts of work beyond what I had previously done in the last project since I already had a createCreature function and the way I had set up battle initially by allowing the user to pick a creature from their vector of previously created creatures transitioned nicely to move from using a vector format to FIFO and FILO linked lists

Assignment4.cpp:

Main Function:

- Declare player1 and player2 lineup objects as well as the loser pile object
- Call createLineup function
- Call tournament function
- List top three winners

CreateLineup Function

- Prompt user to enter how big each team is going to be
- For each team, allow the player to select which character type they want for each character and to give it a name. Instantiate characters of the proper type assigning them to the proper team

Battle Function:

- This function will perform a battle to the death between two characters

Tournament Function:

- This function uses the removeNode() function of the FIFO class to get a pointer to character object from the list
- While both teams have at least one living character, the function will loop, starting battles between the player at the head of the list for each team.
- The winner of a battle is added back into the player's lineup and their score is updated to reflect additional battle points won in the current combat.
- The loser of a battle is added to the loser lineup.
- After combat, if there is at least one live character for each line-up, new characters are selected to fight and the switchStart value is negated to have the other team start combat first.
- Once there are no live characters remaining in one of the lineups, any remaining characters get extra points for making it to the end equal to their current strength points and for the sake of simplicity, they are then aadded to the loser lineup.
- The winning team is then announced based on the total points earned in the tournament.

Character Classes

- Same as previous assignment design

FILO / FIFO Classes

- ADTs updated for a Character *, but really the same functions as in Lab7:
    - void add(Character *)
    - void displayLIst()
    - Character* removeNode()

Trouble During Development / Change of Design / Notes:

- I had the hardest time figuring out how to make the battle sequence end correctly while using the linked list to hold the characters. I couldn't make the class I had designed work for me to end the loop properly once the last item from one of the player's lists was removed. At first I thought I could fix it by having the removeNode function return NULL if there wasn't a pointer to character to return, but that never ended up working for me. Rather I ended up adding a function to the class "endNode". This function returned a bool of whether or not the list was at the head. From here, I was able to include qualifiers for the battle loop of if !endNode, removeNode, else fighter = NULL, where my loop had always been set to break if the fighter object pointed to the NULL pointer.
- It didn't occur to me until late in my designing and implementation that I always had player1 going first throughout the tournament. At that time, I added a bool variable that switched after every battle sequence to change which team initiates the battle.

- I had a hard time figuring out the best way to determine the top 3 finishers. It seems a linked list of this format isn't the ideal scenario to be sorting members of the object the list is made of, so I went with a solution that wasn't incredibly elegant, but got the job done. It seemed the intention of the assignment may have been to have the last three surviving characters as the top three finishers, but after going through the effort to make a scoring system, I really wanted to find a way to assign winners based on number of points the character scored while in play.
    - Within the FILO class, I added a function findWinners() which iterated through the loser pile finding the 1st, 2nd and 3rd place finishers, outputting their scores, names and what player they belonged to.
    - While testing, it came up that if there was only one fighter for each team, the condition in which there were only 2 top finishers (the winner and the loser), the potential prompts and variables for 3rd place needed to be dealt with, so an additional if statement to see if winner3 had been set to something other than NULL was added and fixed the potential problem.
    - **** UPDATE *** After all this, I decided to change my approach again and in the spirit of utilizing the loser pile and it being a FIFO structure, I decided to have any remaining fighters on the winning team fight one another so that a true final winner could be determined fairly. This also got rid of the need to determine any tie break scenarios for players based on tied amounts of points at the end of play. While characters are fighting members of their own team, they are still winning points for their team as well as themselves until the end of play.
- After somewhat jumping into the implementation phase of this project since my set-up seemed pretty much ready to go after assignment 3, I found my battle function getting really long and messy. At a point, I decided to plan and design it out better and break things up into two functions, one battle function for individual characters and a tournament function to keep track of scores and the next players to fight / which lists they should be coming from and going into. This cleaned things up a lot and made more sense with having the code be readable, otherwise there were a lot of while loops and conditionals.
- I worked to remove as much redundant code as I could. I found a lot while going through this process. Rather than processing the battle function based on which fighter had won, I initialized a pointer, winner pointing to the character still alive after battle so that all the prompts and functions associated with the winner only had to be written once.

Testing Plan / Results:

Summary of Testing:

- Testing was hugely important in the development of this program! Unfortuantely I found several of large errors and flaws in my program while

testing. Fortunately, I did a lot of incremental testing, so the errors were typically pretty easy to find and fix, but admittedly, a lot of the errors I had to find came up from changing my mind on my design approaches over and over again throughout this process! There were so many ways to attempt the tournament set-up and scoring that as I was implanting initial designs, ideas that seemed better kept coming up. It was also really good practice with condensing and removing redundant code, but I had to make sure I wasn't accidently introducing errors when doing things like removing if / else statements by doing things like declaring a pointer to character as a winner, rather than having the same logic in the program for two different outcomes based on the fighter, etc;

Test Results Including Incremental Testing:

- Testing between fighting to the death of the different character types was covered last week in my program, so this week's testing was focused on the additional functionality provided by doing a tournament format where the players were being added back into the queue or into the loser list after losing as well as testing the points systems.
    - The battle function was updated to include a heal function and some changes were implemented to condense the battle function, so there were double checks that expected results were being obtained before removing the massive amounts of print statements that end up occurring when a tournament with multiple players is happening.
- Initial testing to esure the Character pointers were being added to the lineups properly were performed with extra print statements in main as well as testing that the loser standings were working properly by printing out the loser list
- Test to ensure restore strength functionality is working
    - Tested by running a tournament with the print statements added to ensure that a proper flip of a coin resulted in adding back half the strength that had been lost during the combat.
- Testing to ensure the switchPlayer functionality is working
    - This was tested with the print statements in the program by ensuring that from battle to battle, it was changing from the name of a character from team 1 to a character from team 2
- Test to ensure that if there's only one team from each team fighting, there isn't an error because the function for finding winners doesn't throw an error since it's looking for 3 fighters
    - Had to fix this!
        - Ended up having to add a bool endNode function to the FILO class so that the loop for printing the top 3 winners could break if there wasn't a node for 3rd place so that the program wouldn't try to remove a node that wasn't there.
- Test that the top 3 ranked players are displayed properly when play is finished with 2 or more players per team fighting

    - Performed as expected though this had to be re-tested after changing the ranking system from being points based to using the FILO structure where ranking ended up being determined solely on order out of play.
- Test that team scores adding up properly by running a small lineup and manually adding up points
    - Found a pretty large error! When implementing the switch player function, I had started down one path where I was going to change which team fighter1 and fighter2 were associated with rather than just changing the battle queue… I had it set up where the wrong players were being drawn for teams and thus team scoring was not working properly. Was extremely happy to have found this error in testing!
    - After this was fixed, tested line-ups of larger sizes and ran multiple times to ensure points were being added up properly and the loser list was being created properly.
    - After being comfortable with all the results seen, I commented out more print statements to make it more manageable for the grader to be able to scroll through the results of a tournament. If desired, these can be added back in.