Adam Kniffin
OSU ID#: 931492308
Email: kniffina@oregonstate.edu

Project 4 Design Description and Testing

Requirements of the Program:

This program is building off of our last project (project 3) so there are certain requirements that are already fulfilled, that do not necessarily need to be restated.

The main goal of this program is to build on to our project 3 to add a tournament style game that a user can play. The user will input how many Creatures they want on their team and then add what Creatures they want to their lineup. Once that has been done each team will fight Creature vs Creature and determine a winner. The winner goes into the winner bracket and the loser goes into the loser bracket. The tournament goes on until a Creature has won. This will be displayed and the top 3 creatures will be displayed.

Test Plan

- For this assignment I plan to use incremental testing. I will start with allowing the user to add Creatures to their team.
    - For this part of the assignment I plan to try adding the amount of Creatures to each list.
        - Once this is accomplished and there are no errors I am going to print the type of Creature and make sure that they were all added correctly.
        - I will use my getType() function to print the string associated with the Creature that was added to my list. If that matches up with the Creature that I added for each team then it will have been done correctly.

Test Results

Part 1 – Adding the Creatures to the correct teams based on what Creature the user has inputted.

- For this part of the assignment everything seemed to work pretty easily with a few minor errors that had to be fixed.
    - I started by prompting the user how many Creatures they wanted to add for each team.
    - Depending on what the user entered I created a new Creature object that was the type that the user had specified.

    - I structured a while loop that would loop until the number of creatures that the user had specified had reached 0.
        - Each time a new Creature was made I decremented the number of creatures until it reach 0.
    - Once I had entered all of the creatures into the correct Stack list I printed all the types of Creatures that were in my respective lists.
- Originally, I expected all of the string types to print, but I kept getting segmentation faults when testing with 1 creature in the list.
    - I saw that it was printing out the “List is empty” when there were not creatures to display so there error had to involve trying to access a pointer that was not there.
    - It turns out that I was trying to accss the head pointer when it was NULL and I had to change it to signify the additional Node that had been created when I first added a new creature.
        - This solved my problem and I was able to move to the next part of the project

Part 2 - Fighting the different Creature from each team and determining who a winner is. Then adding them to the winner and loser brackets accordingly.

- The next step in my testing procedure involved fighting the Creatures from the different lists and then adding them to the winner and loser brackets accordingly.
    - My initial problem was how to retrieve the top Creature from each list. I implemented a function that would return the top Creature on the list so that they could battle one another.
        - When testing this function I kept getting segmentation faults that were similar to when I had errors with my add function.
        - I found the error was that I need to add another ‘else if’ statement in my top function that would take into account when there was only 1 Creature in the list.
- My original idea around testing this was that I would add a while loop that would loop until as long as a Boolean value was false.
    - This would give me easy control over when to exit the while loop.
- Inside the loop I would create a new Creature pointer that would be accessed from the top of each list using my top function that I had described earlier. After fighting, the winner would be displayed and added to the winner bracket, and the loser added to the loser bracket.
    - I would also increment the points for each member variable associated with the Stack’s score integer value.
    - At the end of each round a winner would be displayed and the score would be tallied up and displayed as well.

Results of testing

- Creating the creature pointers and using the top function worked extremely well and I had originally thought would cause a lot more trouble then before.
    - I simply created a Creature object and used a pointer to set it equal to the value at the top of each list.
        - Once I had the value set I could battle the two Creature pointers and move them around accordingly.
- I also had trouble at first because I was not able to filter through my two team's Creature lineups because the top of the list would always be the same unless I either moved it to the back, or just removed it all together.
    - I ended up Creating a Creature pointer and removing the originals from the list. Once the two Creatures fought it would either be added to the winners or loser and the whole team lineup would eventually get to 0.
        - This lead me to my next problem though, how to keep track of the team Score?
            - I decided to add a string to the Creature class to be able to see what team the creatures were a part of.
            - If they won and were on team 1, I would increment team 1's points. The same for team 2.
                - This ended up solving my problem and I was able to continue my problem.

Part 3 – Fighting the Creatures in the Winner and loser bracket and displaying who the winner is.

- Everything up until this point had worked with relatively no huge issues that had to be solved. However, now that I am trying to implement the winner and loser brackets I have to make a decision how I want to keep track of points for each Creature object and team.
    - One of the ideas that I think could work is adding a point member variable to Creature objects and each round the Creatures receive a point if they win, and none if they lose.
        - This would make it so I could keep track of how many points each Creature has and then put the Creatures into ascending order based on how many points they have.
            - I would also have to add a Sort method to the lists which should not be too hard to implement.
    - I am also having trouble trying to implement how the Creatures will be order in the bracket after they have fought.
        - One testing method is to add them to a holding list and then put them back into the winners bracket after their fight.
    - I also need a way to keep track of how many points each team has so they can see who chose the better Creatures.

        - I implemented a string team to all the Creatures and set it to NULL in the constructor of each Creature.
            - Once their team has been determined then they have it set. For later use in the program I will have an if / else statement that allows points to be added to the team that that the Creature is on.

Results of testing

- After testing my above ideas I ran into a couple of problems.
    - 1. Everything worked well for the first round, but in the winners bracket I did not have the tournament until one final winner was chosen, but rather, just 1 round.
        - I need to implement a way so that each round leads to half the amount of Creatures then before, but loops until a winner has been reached.
            - With this idea, I also need to only allow the user to enter 2, 4, 8, 16, etc Creatures because otherwise the Brackets will not add up right.
                - Need the number of Creatures for the user to enter to be a factorial of 2 up to 32 Creatures.
                    - Adding that many creatures became a huge problem, not because the code couldn't handle it, but it just got monotonous and I figured 32 was the most anyone would ever want.
- I also ran into trouble in the Winners bracket. My main problem was figuring out what to do with each of the winners.
    - I implemented a move to back function which sent the winner to the back of the winner's list so that the top Creatures could fight each other.
        - This ended up working well and allowed for the Creatures to be filtered through the winner's list until there was only 1 creature remaining.

My Method for determining the Winner

- When making the program I wanted the game to be based around making the best team. Obviously, someone could just choose Blue Men however many times they wanted to and that team would always win but I wanted to make the game similar to how an actual video game takes place: you fight, and then you regain your health after the fight.
    - With that in mind I chose to restore 100% of each of the Creatures health and physical traits after each round.
        - There is a chance for the underdogs to win.

        - I.E. Medusa gets a Glare special and kills blue men.
    - For the most part though this tournament finds out who the strongest Creature is. It is somewhat of a gladiator style arena where the strongest Creature is left at the end, unless something unforeseen happens (this was done on purpose).
    - I also implemented it this way because I wanted it to feel as though I could add more classes into the game at a later time and have more fair matchups.
        - I.E. I could create a dragon class that could compete with BlueMen.
- With all of that in mind I made the tournament structure pretty simple. Win and you continue, if you lose, you are out and moved to the loser bracket. It is a final four style tournament.
    - 1st Round – You fight the matchup from the other team. Depending on your lineup your Creature will fight the Creature in the same positions as yours.
        - If they win, they get put into the winners bracket, if they loser they are put there.
    - 2nd Round – All of the winners fight against each other. If they win they continue in the tournament, if they lose they are set into the loser bracket.
        - This is the final round and continues until there is one Creature left.
- As I mentioned earlier each Creature is healed for 100% after each round. I could of easily implemented it so that each Creature was healed to 50% after each round or a certain percentage of their total strength points were restored, but I liked the idea of a Gladiator style tournament.

Keep Track of what Creature was in what Standing

- I decided to implement a member variable in my Creature class that would be used to count their points for wins that they receive in the tournament.
    - Each time they defeat an opponent they receive 1 point
    - This would continue until the Creature was defeated. At the end of the tournament there would be 1 Creature left in my winners bracket.
        - This is obviously the winner and I would output this.
    - My Loser bracket contained the rest of the Creatures with different points. There will always only be 1 second place so the Creature at the top of the loser bracket will be second. I also added a thirdplace stack list to put all the third place finishers into after it was determined who they were.
        - In order to find the 3rd place finishers (always more than 1, or a tie) I had to remove the Creature from the loser bracket after it was determined that it was second place.
            - I then would use a Creature point and initialize it to the top of the Loser list. If the points of the next Creature equaled the

points of the first third place finisher, I added it to the third place list.

- This would continue until the points of the original third place creature were higher then the others, or the list was empty.