Sooreechine, Tida
CS162 | Week 8
November 21, 2015

Assignment 4 – Containers

REQUIREMENTS

- Tournament Setup:
  - The tournament consists of two teams of creatures representing two players.
  - The program user will specify the number of creatures both players will use, and each player will choose the order of their fighters to go against the opposing team.
  - Each player will also be asked to specify the names of their fighters.
  - The creatures will fight their opponents using the same set of combat skills (attack, defense, armor, and strength) from Assignment 3.
- Battle Conclusion:
  - At the end of each battle, both fighters will be listed and the winner of the battle is announced.
  - The battle winner will then go back to the end of his team's lineup. A percentage of their lost strengths will be restored via polymorphism while they get rested for future battles.
  - The loser, on the other hand, is disqualified for the rest of the tournament and gets placed in a separate orderly pile.
  - The program will provide an option for the user to display the winner and updated scores for each round.
- Tournament Conclusion:
  - At the end of the tournament, the program will announce first, second, and third place finishers along with their corresponding teams.
  - The program will also list total scores and the name of the team scoring higher points.
  - The program will display the tournament winning team.
  - The program will provide an option for the user to display all the losers in order, with the loser of the first round displayed last.

DESIGN & ANALYSIS

For the lineup of creatures to be fought for each team, I've decided to implement the Queue-like structure similar to the one from Lab 6, due to its First-In First-Out behavior. The first fighter chosen by the player should be the first to enter the field. Similarly, the last fighter chosen would have to wait until all the other fighters have had their turns before they enter the field themselves. Because there are two teams, there will be two sets of singly-list queue for storing each player's lineup of instantiated creatures. Because the game will only be accessing the first Creature object in the order, there should not be a need to traverse down the list, hence the decision to keep the list simple with the a singly-linked list.

In terms of battle between two fighters, the creatures from both teams will fight until one of the fighters goes down with its strength level depleted entirely to zero. The first fight loser will be dropped into a separate pile of dead bodies to be announced later, upon user's request, in the opposite order at the end of the tournament. Due to its First-In Last-Out nature, this pile will be a stack singly-list. The battle winner, on the other hand, will be added back to the back of its original queue, with half of the strength lost in the current battle restored for the future combats. Because there can only be one fighter whose strength reaches zero first, there will be exactly one loser and one winner every battle. This eliminates the problem of having two dead bodies and the order of which they go into the loser pile. A score of 1 is awarded to the winning team.

The first, second, and third place finishers will be represented by the last three characters alive, under the assumption that the user will be specifying at least 2 fighters on each team. In the event that one team is completely annihilated, the three finishers will be those with the highest strengths from that winning team. On the other hand, if the three finishers are of different teams, the main program will traverse the stack of dead bodies and output the name of the second and third place finishers as necessary. This same method will be employed to output the entire list of losers from the pile at the end of the tournament.

Ultimately, the goal of the tournament is to destroy the opposing team. Although the game will keep track of scores of both teams (this will still be displayed as part of the requirements), the true winner of the tournament is the team the first place finisher belongs to.

Codes/Pseudocodes

Assignment 4 will be built upon Assignment 3, Lab 6, and Lab 8.

Queue/Stack Nodes – from Assignment 4 Q&A

```
struct Node
{
        Creature *creaturePtr;
        Node *next;
        Node(Creature *fighter, Node *nextPtr = NULL)
        {
                creaturePtr = fighter;
                next = nextPtr;
        }
};
```

Main Program //tournament

```
  User specifies number of fighters
  Players choose type and name of fighters
        Newly allocated fighters are added to the back of queue
  For loop (from 0 to some large number)      //battle rounds
        If queue1 is not empty and queue2 is not empty
        For loop (from 0 to some large number)      //fight rounds
              If fighter1 is alive and fighter 2 is alive
                    Fighter1 attacks Fighter2
                    Fighter2 attacks back
                    If both are still alive
                          Fighter2 attacks Fighter1
                          Fighter1 attacks back

        //One player is dead
        Winning fighter recovers
        Winning fighter is added to the back his/her queue
        Losing fighter is added to the dead pile stack
  //One queue is empty
  //display the three winners
  //display the dead bodies (from dead pile stack)
```

TESTING

Test Plan & Test Results

Test Case	Input Values	Expected Outcomes	Observed Outcomes
Character choices chosen out of range (Assignment 3)	Input < 1 or Input > 5	Error message is displayed. User is prompted until a correct input is entered.	Worked as expected.
Valid character choices (Assignment 3)	Input = 1, 2, 3, 4, 5	Character of choice is instantiated	Worked as expected.
Valid character names	String input	String input is properly sent to constructor for creature object instantiaton	Worked as expected.
Characters chosen line up correctly. First fighter is the first character chosen by the player.	N/A	Queue addBack() function works properly and character is added to the end of the queue	Worked as expected.
Opposing characters fight multiple rounds until one dies (Assignment 3)	N/A	Opposing characters fight multiple rounds until one of the fighters have zero strength	Worked as expected.
The old winners fight again after other team members have combatted	N/A	Queue addBack() function works properly and character is added to the end of the queue	Worked as expected.
Recovery function restores a percentage of strength points	N/A	Recovery function restores a percentage of strength points	Worked as expected.
All characters fight until one team loses all its members	N/A	The winning team has one or more characters alive. The losing team has zero members left.	Worked as expected.
The losers are announced in proper order. First death is announced last and vice versa.	N/A	The losers are placed in LIFO pile. Losers are pushed into the loser stack and are popped at the end of the tournament in appropriate order.	Worked as expected.
The three winners and team are displayed at the end	N/A	All three winners (dead and alive are displayed.	Worked as expected.
Tournament winner and both scores are announced	N/A	Tournament winner and both scores are announced	Worked as expected.

```
**************************************************
 G A M E  O V E R
**************************************************

Best 3 Fighters:
1. Snape (122) from Team 1
2. Harry (70) from Team 1
3. Umbridge (RIP) from Team 2

Options:
 1. To see the rest of the rankings, press 1:
 2. To visit the Wall of Shame, press 2:
Enter your choice: 1

4. Sirius (RIP) from Team 1
5. Ron (RIP) from Team 1
6. Dementor (RIP) from Team 2
7. Dumbledore (RIP) from Team 1
8. Lockhart (RIP) from Team 2
9. Bellatrix (RIP) from Team 2
10. Dobby (RIP) from Team 1
11. Hermione (RIP) from Team 1
12. Pettigrew (RIP) from Team 2
13. Draco (RIP) from Team 2
14. Voldemort (RIP) from Team 2

---

Player 1's final score = 7
Player 2's final score = 5

PLAYER 1 WINS THE TOURNAMENT!
```

```
**********************************************************************
 G A M E  O V E R
**********************************************************************

Best 3 Fighters:
1. Harry (259) from Team 1
2. Sirius (74) from Team 1
3. Snape (65) from Team 1

Options:
 1. To see the rest of the rankings, press 1:
 2. To visit the Wall of Shame, press 2:
Enter your choice: 2

The Wall of Shame
Displaying the losers in order of their deaths (last to first):
 - Pettigrew
 - Filch
 - Umbridge
 - Dumbledore
 - Lockhart
 - Bellatrix
 - Hagrid
 - Hermione
 - Ron
 - Draco
 - Voldemort

---

Player 1's final score = 7
Player 2's final score = 4

PLAYER 1 WINS THE TOURNAMENT!
```

REFLECTION

This was definitely my favorite assignment by far. I am extremely grateful for all the tortures Assignment 3 and Lab 8 had put me through. I was initially intimidated by how hard those two homework had been, so I was surprised Assignment 4 turned out to be less complicated and more doable than I had imagined. The instruction was fairly straightforward, but it was also extremely tedious and long. Every time I thought I was finished, I would find more requirements I had previously highlighted that were not accounted for. I ended up having a total of 20 files that I had to navigate through, and I was having a hard time keeping all the changes across the creature base and derived classes consistent.

I did not have a lot of time to plan out the pseudocode this time as the scope of the project was extremely large and I was short on time, but I had a very clear understanding of how I would like to execute this assignment. The rules of the game were fairly easy to implement once I got the concepts written down. The first problem I had was not knowing how to put together a linked list of class objects but saw that other classmates also had the same problem in the Q&A forum. Diana Bacon suggested the storing of Creature pointers instead of the actual instantiated objects. I ended up adopting this method myself.

The main deviation I had from the original design was how to choose the last three finishers/best 3 fighters. I did not plan this part very well, as seen above in the design portion, and ended up having to nix the initial idea altogether. I instead chose to focus on the total number of attacks each winner had dealt to his/her opponent

and based their skill points from there. This was the most difficult part of the assignment as I was not sure how to carry out the plan. At first, I wanted to save all the alive fighters into a vector and sort the values within before popping them out in order. Because we are actually dealing with pointers and not the actual objects, I could not figure out how to sort them inside the vector. Instead, I turned to what I assumed was the simplest problem solver and chose to sort the fighters within the team queue itself. Since I already have the removeFront() function done, it was very easy to pop the characters in and out. I ended up getting very lucky and this worked the first time.

Lastly, to account for the polymorphism requirement that I had missed in Assignment 3, I chose to make the recovery() function the pure virtual function. The sub creatures have different methods for strength replenishment, and the Creature class is now an abstract base class.