Kathleen Ann Thurmes
ID# 934039602
CS162: Introduction to Computer Programming II
Project 4: Fantasy Combat Game Part II

Class Diagram

Tournament
Has a DLList

DLList
Has a Node

Node

Battle
Has a Character*
Depends on Menu

Character
Has a Die**
Depends on Menu

Blue Man
Is a Character

Harry Potter
Is a Character

Medusa
Is a Character

Vampire
Is a Character

Barbarian
Is a Character

Die

Menu

Design Descriptions

DLList (DoubleLinkedList)
Node* head Node* tail
DLList() Void deleteDLList() Node* popFront() Character* peekFront() Void addToBack(*Character) Void traverse() Void deleteFirstNode() Bool isEmpty()

Node
Character* chara Node* next Node* prev
Void printValue() Getters and setters

Tournament
DLList team1 DLList team2 DLList losers
Tournament(DLList, DLList) Void runTournament() Void runRound()

Menu
Void mainMenu() Void roundSetup()

Tournament* tournamentSetup() Character* createCharacter(int, string)

Battle
Character* attacker Character* defender
Battle(Character*, Character*) Void runBattle()

Character
Die** attackDice Die** defenseDice Int armor Int strength Bool dead
Virtual int attack() Virtual void defend(int attackRoll) Getters & setters

```
DLList::deleteDLList()
    Until isEmpty() == true,
        deleteFirstNode()

DLList::deleteFirstNode()
    If there are at least two nodes in the list:
        Reassign head pointer
        Delete head’s previous
        Set head’s previous to null
    If there’s only one left in the list:
        Delete the head
        Set head and tail pointers to null

Node* DLList::popFront()
    Node* toReturn = head
    Delete first node ###Ended up having to change this so the node was preserved
    Return toReturn
```

```
Character* DLList::peekFront()
    Return the character from first node

DLList:: addToBack()
    Make tail's "next" point to new node
    Make new node's "prev" point to tail
    Make tail point to new node

DLList::isEmpty()
    Return true if head points to null

Menu:: tournamentSetup()
    Ask user how many characters on each side
    Create said characters and use "addToBack()" of a DLList to store them
    Call Tunament constructor with the two DLLists

Tournament::runTournament()
    While(both teams have players to play)
        Call runRound()
    After, print the necessary ending information

Tournament::runRound()
    Pop the first character from each team's roster
    Pass those into a Battle object constructor
    Run the battle
    Check to see who won (using Character's strength)
    Increment/decrement team scores accordingly
```

Test Tables

Note: character special powers, battle dynamics, Die, DLList, and getNumberBetween have already been tested as part of previous assignments.

Thing Being Tested	**Action**	**Expected Results**	**Passed?**
Characters are being added to linked lists correctly (most of this was already tested in previous labs, so I'm	Add characters to linked list, do some adding, deleting, popping, etc. Traverse list after	Characters will be added, subtracted, and popped as is appropriate	Y

including the short version here)	each action		
PopFront (which I hadn't implemented and tested for a previous lab) doesn't introduce memory leaks and returns the front node	Use popFront() on a DLList with Valgrind. Print the name of the Node's Character to the terminal. Delete the returned node and then the linked list.	popFront will return the appropriate Node with the correct character in it. Deleting the returned node and then the linked list will result in no memory leaks.	Y
peekFront (which I hadn't implemented and tested for a previous lab) doesn't introduce memory leaks and returns a pointer to the character in the front node	Use peekFront() on a DLList with Valgrind. Print the name of the returned character to the terminal. Traverse the list. Delete the list.	The returned character pointer will have the correct name. The first Node will still be in the list when it is traversed. Deleting the list will result in no memory leaks.	Didn't end up needing to implement this
TournamentSetup() allows user to create teams with different numbers of characters	Run tournamentSetup() and give team1 3 characters and team2 5 characters. traverse() both lists	Team1 has 3 characters and Team2 has 5 characters.	Y
TournamentSetup() allows user to create teams with different types of characters	Run tournamentSetup() and give team1 a vampire and a barbarian. Give Team2 a BlueMan and a Medusa. traverse() both lists and print the types of characters they are	Team1 has a vampire and barbarian and team2 has a BlueMan and a Medusa	Y
TournamentSetup() allows user to create teams with duplicate characters	Run tournamentSetup() and give team1 two vampires. Give Team2 two Barbarians. traverse() both lists and print	team1 has two vampires. Team2 has two Barbarians.	Y

	the types of characters they are		
runTournament() stops when one team runs out of characters	Give team1 one barbarian and team2 five BlueMen and run the tournament	Program won't crash from accessing forbidden memory. Tournament is won by team2.	Y
runRound properly determines the winner of the Battle	Give team1 one barbarian and team2 five BlueMen and run the tournament	Team2 wins the tournament	Y
runRound properly adds points at the end of the round	Print out winners, losers, and both team's points at the end of each round	Teams should gain 2 points for each win and lose 1 point for each loss.	Y
Characters recover properly after winning a round	Print out strength of characters right after each round and right before they are put back into their team's list.	Character should gain half of the health points that they have lost since the beginning of the tournament. (not since the beginning of that particular battle).	Y
runRound puts winner into the back of the proper team's list	Print out both team's rosters before and after the round.	The losing character should disappear from the lists and the winner should appear at the back of their team's list.	Y
runRound puts loser into the back of the loser list	Print out the loser list before and after each round	The losing character should appear in the losers list after they've lost.	Y
No memory leaks	Run program with Valgrind. Run three tournaments with different sizes of teams and different types of characters	No memory leaks	Y

Reflection

I struggled with this assignment a bit. Theoretically, it should have been easy to just put another couple of classes on top of what I already had in order to implement the tournament. All the tricky stuff had already been taken care of with the implementation of the 1:1 battles and the implementation of the linked list in our labs. However, I had a nasty bug that only reared its ugly head every once in a while. I would think that I had fixed the problem by changing something around, have it work for a few runs, but then break again after making a change that I thought was completely unrelated.

It took me a while to figure out what was going on, and I changed my original design around quite a bit in an effort to diagnose and circumvent the problem. Hence, why the Tournament class handles all its own setup instead of the Menu class as I had originally planned. Also, there are some awkward and redundant destructor-like functions. I’m keeping them in as a homage to the learning process.

I discovered that in my original implementation, a vampire versus a vampire battle could go on forever (or until I got a memory error). This was because I had seeded the random number that was used to determine whether to use “Charm” or not every time the defense function was called. The program was running quickly enough that the re-seeding happened over and over again at what was functionally the same time and, therefore, the same seed. If it happened that the seed resulted in the vampires both using “Charm” over and over again, the vampires would just sit there and charm each other until my computer ran out of memory. But, of course, that only happened half the time (less than half the time, even, because sometimes the clock seed would change partway through the run).