Harrison Latimer

CS 162

Assignment 4

Program Outline + Test Procedure

Identify the problem: Since most of this project builds off of Assignment 3 the primary problem will be coordinating the dynamic linked list of member class variables.

Modularize the Solution: With inheritance this will prove to be a much simpler task. The Character class will be inherited by all five derived class objects to reduce redundant code. Another benefit to this base class is the pure virtual functions that can easily be adapted in bases classes to meet the special requirements of each of the derived class characters.

Each class will have a default constructor to set its protected variables and two functions to deal with attacking and defense. For these two functions the primary difficulty will be producing random events and dealing with battles of characters of the same type. Also finding a way to deal with Medusa's glare that is at the character class level and not inside a Menu function.

Through dynamic memory allocation and dynamic linked lists the teams can easily have characters added or removed during game play.

Barbarian vs Barbarian

Functions	Test1	Test2	Test3	Test4	Test5
Barbarian1.attack()	3	7	6	7	5
Barbarian2.defense()	3	9	7	4	7
Barbarian2.atttack()	4	5	7	10	8
Barbarian1.defense()	4	4	5	11	4
Errors?	Random generating of attack / defense produces the same values		Used an abs function for defense calculation and added to defense rather then subtracting		
Solution?	Use chrono library and seed random num generator		Remove abs from defense calcualtion		

Barbarian vs Vampire

Functions	Test1	Test2	Test3	Test4	Test5
Barbarian1.attack()	5	7	8	10	n/a
Vampire.defense()	4	5 – special attribute triggered	3	3	n/a
Vampire.attack()	7		6	7	n/a
Barbarian1.defense()	3		5	5	n/a
Special Ability	--	Charm – worked fine	--	--	n/a
Errors?				Vampire was dead but still attack a last time	n/a
Solution?				Added check to selection statement in menu class	

Vampire vs Blue Men

Functions	Test1	Test2	Test3	Test4	Test5
Vampire.attack()	5	7	4	3	9
BlueMen.Defense()	10	6	8	12	4
BLueMen.attack()	12	6	11	9	8
Vampire.defense()	6	4	6	3	4
Special Ability	--	BlueMen lost a defense die	--	Vampire used charm	
Errors?					Blue men lost 1 die instead of lose two
Solution?					Alter selection

					statement in bluemen class

Harry Potter vs Medusa

Functions	Test1	Test2	Test3	Test4	Test5
Harry Potter.attack()	5	6	3	5	7
Medusa.defense()	4	3	4	2	3
Meduse.attack()	6	12	5	6	3
HarryPotter.defense()	4	3	6	5	5
Special Ability	--	Medusa kills harry potter. Harry potter comes back but with 10 health			
Errors?		Harry potter should be brought back with 20 strength			
Solution?		Edit special attribute in harry potter class			

Dynamic linked list test

Functions	Test1	Test2	Test3	Test4	Test5
Harry Potter.attack()		6	3	5	7
Medusa.defense()		3	4	2	3
Meduse.attack()		12	5	6	3
HarryPotter.defense()		3	6	5	5
Special Ability		Medusa kills harry potter. Harry potter comes			

		back but with 10 health			
Errors?	Program exits without any battle	Harry potter should be brought back with 20 strength			
Solution?	Characters were not pushing to the linked list. Corrected a syntax error	Edit special attribute in harry potter class			

Reflection

Coordinating special abilities across different derived classes and generating truly random attacks and defenses were the main issues that needed to be resolved during the testing phase. Overall the implementation of the derived class with pure virtual functions was not much different then having virtual functions in the base class.

Making the linked list was fairly straight forward and the issues I did have were due to either syntax error or memory deallocation issues. These were straight forward to solve through debugging and IDE flags.

The time to plan, write code, and test was the quickest cycle so far for 162. With a set of menu functions and input validation functions written for other projects it has made it easier / faster to get a program up and running.