Derek Yang

Document (Design + Reflection)

Program Design

Problem: Create a fantasy combat game where there are separate characters each with their own individual special characteristics. These characters will inherit from the parent characters class and use polymorphism for their own special attacks. Then implement a tournament with your own structure type, cannot use standard containers. The following are necessary conditions of the game:

- Create the following characters: vampire, barbarian, blue men, medusa, and harry potter.
- Each will have it's own attack and defense rolls. The output depends on the number of sides to the die and the number of rolls.
- Each character will have it's own specific armor and strength points value. Strength points is like HP in regular games.
- Each character will have it's own characteristics or special skill. This skill will be created using polymorphism of existing or new classes. Usually this will mean that the parent class will have an abstract class.
- There will be a character base class
- The menu should display the attacker and defender's information
- The tournament will be held like a lineup for each team.
- When one character dies, they are moved to the loser pile
- Once in the loser pile, the program should print the losers reversely at the end of the round
- If a character wins against another character, the character is given some strength points back
- The user should be able to name each character

Character class – parent
Variables: attack, roll sides, roll turns, defense, strength

- Attack function
- Defense function
- Special ability abstract class
- Gets and sets for the variables

Vampire class – child

- This class will have the special ability charm
- Charm will activate during the defense phase and reduce incoming damage to 0
- This gives the effect of the skill activating

Barbarian – child

- This class will just inherit from the character class.
- Not much is changed in this class, it can be counted as our control class

Blue Men - child

- This class will activate mob during the defense phase
- Mob must take the factors of 4 in order to work correctly.
    - You can not use the incoming attack and divide it by 4 because the overall strength will not be used. You must take the overall strength and reduce the number of dice from there.

Medusa – child

- Medusa has a special attack called glare
- Glare activates during the attack phase
- The only extra consideration for glare is that it will activate Hogwarts and harry potter will survive if it is activated the first time

Harry Potter – child

- Harry potter has a special ability called Hogwarts where the user's strength points is set to 20 if the user is supposed to die
- This correlates with medusa's glare ability and will only activate once.
- Use a bool tracker for this special ability
- Activate during the defense phase of the character's turn

Game Class

- This will play the game
- Loop through each round and ask the user to see which character to pit against
- The round will end when a character dies
- Each round, the game class will take the attack and defense options of the characters and feed them into each other. The receiving end will be the defense character.
- The game class will keep track of the number of rounds.
- The game class will also

Dice Class

- This class will implement a dice which will take in the number of sides and turns and return the result
- The dice class will also print each roll and random number in the class itself to keep the program clean

doubleLink Class

- This class is used for the loser pile
- The loser pile will be added in a list and the result will be printed
- The class will need to clean up the new character objects

Queue Class

- This is a circular linked list
- This class will be used to keep the lineup
- The winner will be moved to the back of this class and have their strength replenished

Main

- This should only have two or three functions to play the game

Input Validation

- The user will be asked mainly integers in the menu options. The cases are usually within a range of answers. Therefore, I will create a function that checks for an int, and then if that int is within a specified range.
    - Ex: choose 1 to 3 will ask for an int within range 1 to 3
    - Ex2: choose starting grid width will be min 1 and max int

Test Case	**Input Values**	**Functions**	**Expected Outcomes**	**Observed Outcomes**
Input too low	Input is less than the min	Main() Validation() If input < min	Loop back and ask the user to re-input	Continually asks user to enter a higher int
Input too high	Input is more than the max	Main() Validation() If input > max	Loop back and ask the user to re-input	Continually asks user to enter a lower int
Input in correct range	Input is within max and min	Main() Validation() If min>input>max	Accepts the correct answer and exits the while loop	Loop stops and accepts correct int
Input is not an int	Input is a string, bool, float or other variable	Main() Validation() If input != int	Loop back and ask the user to re-input an int	Loops until an int is entered within the designated range.
Medusa glare activates for harry potter	Medusa and harry potter chosen	Main() Harrypotter() Medusa() Game(0	Harry potter will not die the first time but rather the strength will be set to 20. If the skill glare activates a second time it will cause the user to die.	Harry potter dies the second time but not the first time. Strength points is set to 20. Medusa usually ends up winning if glare is activated.
Vampire uses charm as defense	50% of the time vampire will nullify any damage	Vampire() All other classes() Game()	Vampire will not take in any incoming damage or calculate it into the strength points	The damage value is not using during vampire's defense phase and the strength points are untouched when the skill is activated
Blue men uses mob	Mob activates and the number of	Bluemen() All other classes() Game()	Bluemen will activate mob if	The degree of factor of 4 is used for the total strength. Each time mob is

	dice is reduced		the dmg exceeds 4	activated, the blue men will reduce the number of defense die turns. This also only happens if the total strength is effected and not only the raw attack number. Otherwise there will be a lot of residual left over.
Attack	The character attacks	All child character classes() Game()	Character attacks and returns the attack to the game class. The game class will then use this and pass through the defender	The defender receives the results of the attacker's attack number from the game class. If the attacker is medusa and glare is activated, the attack value will be very large to ensure instant death
Defense	The character defense and receives the attack as input	All child character classes() Game()	The character defends and receives the attack information for the other character from the game class.	The defender receives the attack information from the game class and calculates the information accordingly. Bluemen, vampire, and harry potter will need to implement special functions in order to use their abilities.
User enters 1 for team size	The user enters 1 max team size	Double linked list, circular list, and gameplay with character classes	The double linked list and circular list will only reference the head because there is only 1 character each	The head is referenced and there is only one character displayed per data structure. Ex: one in linked list and one in circular list
User enter 1+ for team size	The user enters 1+ max team size	Double linked list, circular list, and gameplay with character classes	The double linked list and circular list will need to use more than the head to store the characters	The head is used first and then each new character is added to the front of the queue. For the loser list, all characters are added to the back of the list, but it is printed reversely.

Reflection

In this program, we had to pit two different characters up against each other in a tournament and ask the user if they would like to continue playing the game. To start, we had to combine a few of our past projects and labs into one assignment. The easy part was combining both the labs and the assignment into one program. I found that originally, I had planned only to use the circular linked list to store the characters. I quickly found that I needed to implement a doubly linked list as well to store the loser list. Making the program work was easy at first. The main issue which I ran into which was not accounted for in my design was the memory leaks. I thought to myself that I could just do without checking for memory leaks during the programming process. Boy was I wrong. I spend hours looking for the memory links between the doubly linked list and the circular list. I even tried to modify and delete character objects from nodes inside of my gameplay class. This did not work out because I was not actually deleting the pointer. After many hours of looking for the memory leaks, I started to look at valgrind's output and see why we received certain functions which contained these leaks. I had to test one by one and line by line for the doubly linked list and circular list in order to delete these classes. I really started to get the term "divide and conquer" this really helped motivate me to finish the project because at first it was overwhelming looking at all the leaks.

Troubleshooting the leaks, I decided to start with one end and continue to the other. So I started with the circular linked list because it was easier for me. I started to put delete in random areas to delete the character classes however I found that after doing so, I was able to get rid of half the leak. This did actually help me to understand where to put the delete later on when I needed to troubleshoot the doubly linked list class. In a scramble, I started to place random check functions and std::cout statements everywhere. I now know that this is not a good practice as it can just make the code look unreadable and more confusing. I found that if I added a keyword that is searchable so I can delete the items later it would make things easier. I also commented out the extra character information which was needed in the last project but not needed in this one (such as character strength, defense, etc). I left the statements in for the TA so that way it is easier to check later on.

Overall I found that this project taught me that it is easy to put things together but it is hard to make them mesh. The memory leak aspect of the project really made me think about how much I should really pay attention and slow down when I code. It is not necessary to keep coding to just finish the function, finishing the function does not finish the project. The design itself felt like a puzzle because I was just using my old code as current containers. Since I started to practice with vectors beforehand, it made it easier to implement my own code within the project.

Class Hierarchy:

Character – parent class

- Int strength, defense, number of sides of die, number of rolls of die, special attack
- String description, skill description, name of character
- Getter's and setters for all variables
- Virtual functions for attack and defense

Vampire – child class

- Inherits from the character class
- Has special ability charm which negates all incoming attacks
    - This ability will polymorph the defense function

Medusa – child class

- Inherits from the character class
- Has special ability glare which maximizes attack output for an instant win
    - This ability will polymorph the attack function

Blue Men – child class

- Inherits from the character class
- Has the special ability mob which will reduce the number of defense rolls when the strength is modified by 4
- This ability will polymorph the defense function

Harry Potter – child class

- Inherits from the character class
- Has the special ability Hogwarts which revives the character and sets them to 20 strength after revival
- This skill will modify the defense function and have a bool flag to mark if the ability has been used.

Barbarian – child class

- This will be a direct inherit from the character class and will not be modified except for the base parameters.

Dice, Gameplay, Menu, Validation, Queue, QueueNode, doubleLink, and Node do not inherit.