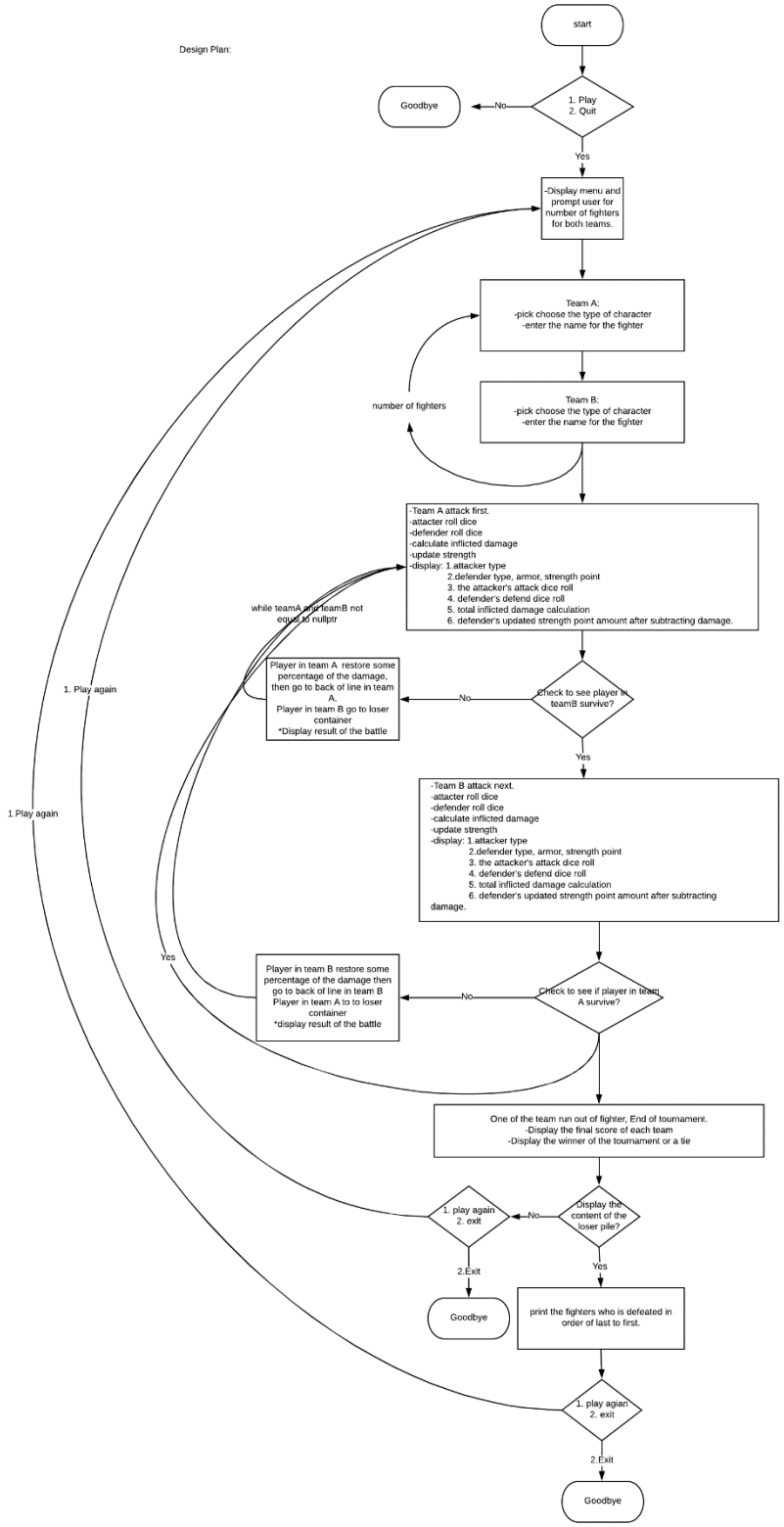
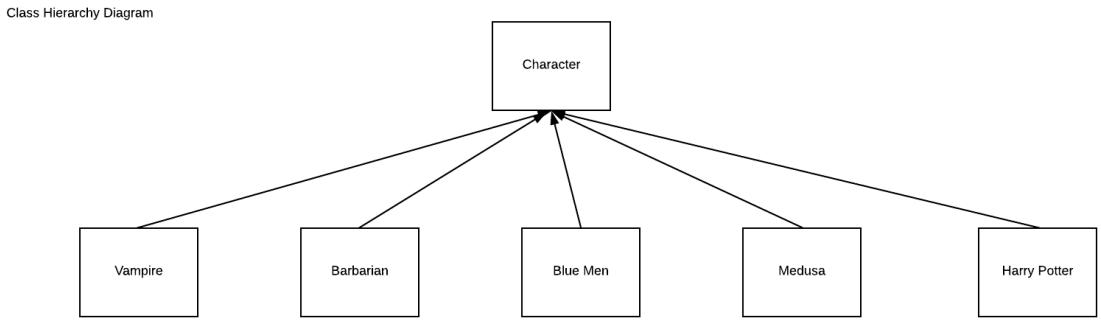
Design Plan:


start
1. Play
2. Quit
No
Goodbye
Yes
-Display menu and prompt user for number of fighters for both teams.
Team A:
-pick choose the type of character
-enter the name for the fighter
Team B:
-pick choose the type of character
-enter the name for the fighter
number of fighters
-Team A attack first.
-attacter roll dice
-defender roll dice
-calculate inflicted damage
-update strength
-display: 1.attacker type
2.defender type, armor, strength point
3. the attacker's attack dice roll
4. defender's defend dice roll
5. total inflicted damage calculation
6. defender's updated strength point amount after subtracting damage.
while teamA and teamB not equal to nullptr
Check to see player in teamB survive?
No
Player in team A restore some percentage of the damage, then go to back of line in team A.
Player in team B go to loser container
*Display result of the battle
Yes
-Team B attack next.
-attacter roll dice
-defender roll dice
-calculate inflicted damage
-update strength
-display: 1.attacker type
2.defender type, armor, strength point
3. the attacker's attack dice roll
4. defender's defend dice roll
5. total inflicted damage calculation
6. defender's updated strength point amount after subtracting damage.
Check to see if player in team A survive?
No
Player in team B restore some percentage of the damage then go to back of line in team B
Player in team A to to loser container
*display result of the battle
Yes
One of the team run out of fighter, End of tournament.
-Display the final score of each team
-Display the winner of the tournament or a tie
Display the content of the loser pile?
No
Yes
print the fighters who is defeated in order of last to first.
1. play again
2. exit
2.Exit
Goodbye
1. play agian
2. exit
2.Exit
Goodbye
1. Play again
1.Play again



Before I started this project, I though this project will be easy to do. I will use what I wrote for project 3 and Lab 7, then add stack class for the loser container. However, it took longer than what I expected to modified lab 7 and write the stack class because it is not just a simple integer anymore.

I used a print function to print out what is in the queue before adding, after adding and before deleting and after deleting. This helps me to see what step in the process didn't go the way I expected. I didn't have any issue just one player in each team. However, when each team has more than two players, I was having segmentation fault. I read each step carefully and realized the problem is in my addBack function, which leads to my removeFront function. I had to change how my removeFront function works in order to have my addBack function work.

This process took me a long time. After this problem was resolved, everything seems to work but there were some memory leaks. I inserted some statement in my destructor and realized my destructor didn't run at all. I realized I didn't not delete my queues and stack. After adding the delete, the program was working until the very end. I couldn't print out the loser list anymore. I looked over the codes again and realized I deleted the loser container too early. After deleting in the proper place, the program is running well again.

To check if my recover function is working properly, I print out the new recover strength. I had the player recover 100% initially, so I know if the player is recovered correctly. Once the I have the recover function working properly, I change the recover function. The winner will get back 50% of damage to the reminding strength instead of 100%.

Run the program to make sure the score for each team are updated correctly and the score should reset if user wants to play again.

Segmentation faults and memory leak are my weakness in this course. Every time I think I understand something, something else go wrong. I am still trying to find a method that will help me understand the concept better. Do you have any recommendations? I understand the basic and general concept, but whenever it involved with pointer of objects, I still gets confused sometimes.

I will be using my bonus day for this project. I couldn't work on this project because my twins were sick. Luckily, I have not use any of my bonus days yet.

enter	expected
One player per team	One player per team A, One player per team B
Zero player per term	Invalid input
Six players per term	Invalid input, max 5 players per team

	Expected
If teamA point > teamB point	Team A won
If team B point > teamA point	Team B won
If teamApoint = teamB point	tie

Winner per battle	Team +2 points	Points updated as expected.
Loser per battle	Team +1 points	Points updated as expected

When player win the battle	Expect to recover 50% of damage, remove from front of line up and add to back of lineup
When player lost the battle	Expected to be removed from lineup and add to the loser stack.