Fantasy Combat Tournament

Goals

- Develop containers to hold characters using linked list structure
- Modify an existing parent abstract class

In this project, we will develop the game from project 3 to run a tournament between 2 teams of fighters. If you have issues with project 3 and need to begin fresh, reference the character descriptions from project 3 to build your characters.

Requirements

Gameplay

This game is a **one-user-two-player** game, so one user should be able to play setup fighters for both teams. For simplicity, we will call the order of fighters in each team “lineup”, like a batting order in baseball or softball.

The **flow of gameplay** is as follows:

At the start of the tournament, it should prompt the user to **enter the number of fighters** for both team. After that, for each fighter, the game should ask the user to **choose the type of character, and enter a name for the fighter**. For example: assume the user chooses Harry Potter for player No.1’s team lineup, the user can name the Harry Potter character “Larry Snotter” or just “Harry Potter No.1”. The game should allow more than 1 of the same character for team lineups. For example, Team A has 3 members: 2 barbarians and 1 Harry Potter. The order of the lineup should be the same as the order user entered.

After the user supplied the lineup of fighters in order, for Team No.1, and Team No.2, the tournament starts. In the tournament, the **fighter at the head/front of each lineup will battle**, in the same way they fight in project 3. The winner of each fight gets **put at the back of their lineup**, while **loser goes to the top of the loser container** (There should only be one containers to hold defeated fighters on both team).

If a fighter won, it may have taken damage, so the game should **restore some percentage of the damage** they took when they get back in line.

The lineup order cannot be changed, meaning that the game cannot modify the order of the lineup except when putting winner at the back of lineup and loser at the top of the loser container.

The tournament finishes when one of the team does not have any more fighters to fight. When the game ends, it should **print the result of the game**, and allow user to **have the choice to display the contents of the loser pile**. More information is shown below in the menu section

Containers

For this game, you need to make your own containers for the team lineups and the loser containers. It is required to use **stack-like or queue-like linked-list structures** for the container.

Important: Please **make your own containers** instead of the STL containers.

Recovery

For recovering function for the winner of each combat, make it simple by adding a member function to the parent class. For example: a random percentage by rolling a die (a roll of 5 means 50% recovery), or some other schemes of yours.

Note: If you would like to implement a different recovery function for each character, that is fine.

Menu

At the start of the game, the menu should display 2 choices for user:

1. Play
2. Exit

If user chooses to play the menu should prompt user for number of fighters for both team, and ask for the type of character each fighter is, and the name for each fighter. From this point on the game does not require any more user interactions until the tournament is over.

During the tournament, the game will output the information about each battle in the tournament. For each fight between 2 teams, the game should display the type of character and the name of 2 fighters, and which won the combat.

Ex: Round 1: Team A Blue Man No.1 vs. Team B Harry Potter No.1, Harry Potter No.1 Won!

After the tournament is finished, the game should **display the final score for each team**, and then, depending on the final score, **display the winner of the tournament**, or a **tie**. The scoring system is your design decision, for example,

winner +2pts, loser -1pts. Afterwards, provide the user a choice to whether display the content of the loser pile. If the user chooses yes, print them out from top to bottom (fighters who is defeated in order of last to first).

After the results are displayed, the menu should provide 2 options for user to choose:

1. Play again
2. Exit

If the user chooses to play again, the who process of choosing fighters and the tournament restarts. Otherwise, the game exists.

Character class

The requirement should be the same to project 3, where you should use polymorphism, and all characters should inherit from the **Character class**. Characters class should be an abstract class.

To hold all types of characters in the containers, each object should be instantiated and their pointers should be put into the lineup, so the containers should contain only Character pointers. If you are uncertain how to implement the lineup with Character pointers and using polymorphisms, ask questions on Piazza or go to one of the office hours.

Reflection Document

The requirements for reflection document have been specified in the Project 1 document.

Make sure your document has **design descriptions, test tables, and reflections**. The reflections include: changes in design, problems encountered, and how you solve those problems. Additionally, the reflection document should include a **class hierarchy diagram**.

Important Note

Note that this project uses project 3 as a starting point. Since we could not release a sample code of project 3 now because some of you will turn in it late, so you have to figure out how to implement those details based on project 3 by yourself. But you will not be penalized twice.

What you need to Submit

- All the program files including header and source files (.cpp/.hpp)
- makefile
- Your reflection pdf file

Important: Put all the files in a single .zip file and submit it on Canvas.

Grading

- Programming style and documentation: 10%
- Create tournament, get the fighters, resolve the round, and put the fighters into the appropriate containers: 25%
- Prompt the user to enter the number and type of fighters and fill the two lineups based on the number and type entered by the user: 10%
- Create and properly use the structures to hold the lineups: 10%
- Create and properly use the structures to hold the loser pile: 10%
- Create a recovery function for the winners of each round: 10%
- Implement a system to determine which side won the tournament and display the results: 10%
- Reflections document to include the design description, test plan, test results, and comments about how you resolved problems during the assignment: 15%