Project 1: The New Normal in Middle-Earth

CMPE 250, Data Structures and Algorithms, Fall 2020

Instructor: H. B. Yılmaz

TAs: Umut Kocasarı, Rıza Özçelik

Due: **November 27, Friday, 23:55**

1 Introduction

There is definitely something wrong with the world nowadays. Lots of lots of strange incidents, one after another... Not only our world, though, Middle-Earth[1] is also going through a rough patch.

Orcs have gone away and the others have fallen out with each other. They have formed small communities and the wars have begun. Communities attack one another, they try to kill the rival hobbits and capture more territories. The future of the Middle-Earth is desperately gloomy. But, you can help them, you can end this freaking, useless, lose-lose turmoil! You can lead the people of the Middle-Earth out of the darkness, right towards the bright horizons!

"What can I do?", you asked? Simple! Just predict the "winner" of the fights even before they began. They would listen to you and stop the fight; you are a true leader!

This is a difficult mission to undertake and we will help you as well. Together, we will end this non-sense the only way we know how: coding! We already started to work, but desperately need you to complete the key parts.

2 War Rules

Even wars have rules, and this also holds in the Middle-Earth. Here they are:

- Middle-Earth hosts five different character types: Hobbits, Men, Elves, Dwarfs, and Wizards. Each community contains exactly one of these characters; 5 characters in total.
- Only two communities involve in a war and they attack each other in rounds. The attack is performed by a character of the attacking community towards a character in the defending community. For instance, in round 1, the Wizard of community 1 can attack the Dwarf of community 2 and in round 2 the Men of community 2 can attack the Hobbit of community 1. As a convention, community 1 attacks first.

[1] Wikipedia page of Middle-earth

- Each character has predefined attack and defense points. The damage of an attack is computed by subtracting the defense point of the defender from the attack point of the attacker. If the damage is larger than 0, the health of the defender decreases by that amount. When the health of a character drops to 0 or below, the character is dead. As an edge case, negative health scores are considered as 0.
- If a dead character tries to attack or defend, it is replaced by the name-wise alphabetically next alive character in its community. If there is no alphabetically next alive character, then it is replaced by the alphabetically previous alive character.
- Hobbit is the most important character type. If the hobbit of a community dies, the community loses the war. In addition, if all other community members die, Hobbit becomes defenseless and the war is lost again.
- The maximum number of rounds a war can last is predefined. If none of the end conditions is met when the round limit is reached, the war ends in a draw.
- Some characters have special skills that they might use per certain number of rounds.
  - Elves can convey half of their health (integer division) to the hobbit of its community per 10 rounds. In other words, at least 10 rounds must pass for an Elf to trigger its special skill, since the beginning of the war and the last use of the skill. After using the special skill, Elves can still attack the defender.
  - Dwarfs can call another dwarf with its own attributes and they attack defender together. This doubles the damage on the opponent for this round and then the callee disappears. Dwarfs can trigger this skill per 20 rounds.
  - Wizards can turn the last killed community member back to life with the initial health of the killed member, per 50 rounds. After the resurrection, the wizard still attacks the defender. If all members of the community are alive, the special skill is not triggered at all, so that the wizard would not have to wait for 50 rounds.

The rules are set! All you need to do is to answer the questions below before the war starts. Hurry!

Winner Who is the winner? "Community-1", "Community-2", or "Draw"?

Number of Rounds How many rounds did the war last? Remember, rounds start from 1.

Number of Casualties How many characters would die if the war actually had happened? Let them know how many lives you saved.

Health Records How would the health of the characters would progress during the war? Let them know how a war would make them suffer.

Now let us discuss what we already did for you.

3 Starter Codebase

We created three files: `Character.h, Character.cpp,` and `main.cpp`, where you will need to edit only `Character.cpp` and `main.cpp`. In other words, `Character.h` is already completed for you and you will fill in the methods in `Character.cpp` to create the characters. Besides, you need to edit `main.cpp` to perform I/O operations and simulate the wars.

3.1 `Character.h/cpp`

This class consists of the attributes of characters, constructors, operator overloaders, and destructor. Even though you might not use all the class attributes and functions depending on your code, you **must** fill every one of them. Let us describe each class attribute:

isAlive Whether the character is alive.

name A unique identifier of the character. Can be any string, like Frodo.

type Type of the character. One of Hobbit, Men, Elves, Dwarfs, Wizards.

remainingHealth The remaining health of the character. The initial value will be provided as input.

attack Attack point of the character.

defense Defense point of the character

nRoundsSinceSpecial How many rounds have passed since the special skill has been used.

nMaxRounds Number of maximum rounds in the war.

healthHistory A pointer to record the health history of the character for each round. The history should record the health before and after the war as well, which means if a war lasts 100 rounds, this variable should store 101 values.

3.2 `main.cpp`

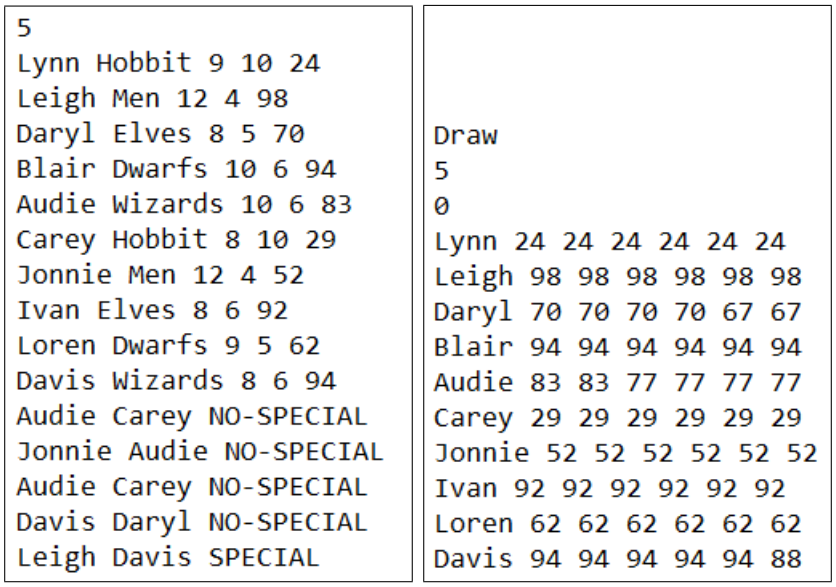
This is the class where you will be handling the input, output, and war simulation. You should use the `Character` class, its methods and combine it with the simulation code.

4 Input & Output

Your code must read the name of the input and output files from the command line. We will run your code as follows:

```
g++ main.cpp Character.h Character.cpp -std=c++11 -o project1.out
./project1.out inputFile outputFile
```

Make sure that your final submission compiles and runs with these commands.



```
5
Lynn Hobbit 9 10 24
Leigh Men 12 4 98
Daryl Elves 8 5 70
Blair Dwarfs 10 6 94
Audie Wizards 10 6 83
Carey Hobbit 8 10 29
Jonnie Men 12 4 52
Ivan Elves 8 6 92
Loren Dwarfs 9 5 62
Davis Wizards 8 6 94
Audie Carey NO-SPECIAL
Jonnie Audie NO-SPECIAL
Audie Carey NO-SPECIAL
Davis Daryl NO-SPECIAL
Leigh Davis SPECIAL
```

```
Draw
5
0
Lynn 24 24 24 24 24 24
Leigh 98 98 98 98 98 98
Daryl 70 70 70 70 67 67
Blair 94 94 94 94 94 94
Audie 83 83 77 77 77 77
Carey 29 29 29 29 29 29
Jonnie 52 52 52 52 52 52
Ivan 92 92 92 92 92 92
Loren 62 62 62 62 62 62
Davis 94 94 94 94 94 88
```

Figure 1: Sample input (left) and output (right) files

4.1 Input

All input files have the following format:

The first line states the maximum number of rounds this war can last and the next 5 lines list the characters of the first community. Each of these lines comprise two strings and three integers to denote the character *name*, *type*, *attack point*, *defense point*, and *initial health*, respectively. The next 5 lines contain the same information for community-2, in the same format.

The community descriptions are followed by *maximum number of rounds* lines. Each of these following lines specifies the attacking move in the corresponding round by three strings: *name of the attacker*, *name of the defender*, *whether the attacker triggers the special skill or not*. Note that the attacker cannot trigger a special skill arbitrarily, even if it tries to do so: a number of necessary rounds must pass to reload the skill and the character must have a special skill in the first place!

Below is the input template and Figure 1 displays an example input on the left.

```
max_number_of_rounds
community1_char1_name type attack defense health
community1_char2_name type attack defense health
community1_char3_name type attack defense health
community1_char4_name type attack defense health
community1_char5_name type attack defense health
community2_char1_name type attack defense health
community2_char2_name type attack defense health
community2_char3_name type attack defense health
community2_char4_name type attack defense health
community2_char5_name type attack defense health
attacker_name defender_name attacker_name defender_name if_special
attacker_name defender_name if_special
attacker_name defender_name if_special
```

Round-1	Audie attacks Carey without special skill. Audie's attack point is 10, and Carey's defense is also 10. So, there is no damage given to Carey.
Round-2	Jonnie attacks Audie. Jonnie's attack point is 12 and Audie's defense point is 6. 6 damage is given to Audie, which is reduced from her health.
Round-3	Audie attacks Carey. Audie's attack point is 10 and Carey's defense is also 10. So, there is no damage given to Carey.
Round-4	Davis attacks Daryl. Davis' attack point is 8 and Daryl's defense is 5. 3 damage is given to Daryl, which is reduced from his health.
Round-5	Leigh attacks Davis. Leigh's attack is 12 and Davis defense is 6. 6 damage is given to Davis, which is reduced from his health. It is normally a special skill round but men do not have a special skill. So, there is nothing different than a NO-SPECIAL round, in practice.
Query-1	First query is who the winner is. It is draw because none of the two end conditions occurred.
Query-2	Next query needs us to print the number of rounds passed. 5 rounds have passed in total, and we print 5.
Query-3	Next query asks the total number of casualties. None of the characters is dead, so it is 0.
Other Queries	Next queries are to display the health records of characters round by round.

Table 1: Explanation of the expected output file

```
attacker_name defender_name if_special
attacker_name defender_name if_special
attacker_name defender_name if_special
attacker_name defender_name if_special
⋮
⋮
```

4.2 Output

The output file must consist of 13 lines. The first line must state the winner: Community-1, Community-2, or Draw.

Each of the second and third lines must contain one integer to denote the number of rounds passed and total number of casualties, respectively.

The remaining 10 lines must have the same format. They must contain the health record of each character in the order they were presented. Each line must start with the name of the character and must be followed by a space-separated list of integers. Keep in mind that the size of the health records is equal to the number of rounds the war lasted plus 1.

Figure 1 illustrates the output for the sample input and Table 1 describes the war round-by-round for the same file.

5 Grading

We will automatically grade your submission with multiple test cases besides the ones one we already provided. So, **ensure that your submission is runnable with the commands**

we provided above and satisfies all project requirements. The auto-runnability of the submission will constitute**10/100** of your final grade and passing the test cases will earn you **90/100** more points. If your submission is not auto-runnable, then you will receive 0 at first and then have to issue an objection.

Warnings

1. You are allowed to use the standard C++ library, but not additional libraries. All requirements of the project must be satisfied with your authentic code, not by any external code. **Otherwise, you will get zero**.

2. You can edit `main.cpp` as you like, however it is not allowed to change `Character.h` and you should change only the body of the functions in `Character.cpp`, not the headers.

3. You are expected to implement the copy constructor and the destructor, and overload the assignment operator. Otherwise, when extensively tested, your code will very likely give segmentation fault and crash even though the war simulation is just fine.

Submission Details

You are supposed to use GitHub Classroom for the submission. No other type of submission will be accepted. Also, pay attention to the following points:

- All source codes are checked automatically for similarity with other submissions and exercises from previous years. Make sure you write and submit your own code.

- You are expected to use C++ as powerful, steady, and flexible as possible. Use mechanisms that affect these issues positively.

- Make sure you document your code with necessary inline comments and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long. This is very important for partial grading.