MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG315 Algorithms

Assignment 1: A Game Leaderboard System

This assignment aims to help you practice the **heap data structure** and **heap sort algorithm**. Your main task in this assignment is to develop a simple leaderboard system **using C programming language**.

Overview

Every leaderboard needs a couple of sorting features to help you see which participant is the leader when it comes to a certain aspect. Having the most wins does not mean the best win rate, and the expectation to win placed upon each opponent plays a major role, especially in the betting department. Since heap sort has been the highlight of the week, the focus will mostly be on implementing it to sort our leaderboard the way the user desires.

In this assignment, you will develop a leaderboard system for an online battle arena video game which supports the following functionalities:

1. Read a data file containing the names of every champion and how much people bet in their favor, which for the sake of simplicity, we will consider to be their expected win rate.
2. Read a data file containing information about every battle that took place between said champions. The information is basically the identity of the participants and who won.
3. Sort or rank every champion based on either their expected win rate, actual win rate, or expectation skew.
4. Print the results of the sort with the respective information.

Input

The program will have three inputs:

- Sorting criteria: (1) actual win rate, (2) expected win rate, or (3) expectation skew.
- Champion Data: A file will contain the names of every champion and their expected win rate separated by a space (i.e., championName expectedWinRate) in which each line will correspond to a different champion. Do not make any assumptions about the number of champions. championName can have maximum 50 characters without any space.
- Battles' Data: A distinct file will contain information about every battle that took place between these champions. Each line represents a separate battle and every piece of information in a line is separated by a space (i.e., battleID battleParticipant1 battleParticipant2 winner).

This program takes these inputs them **as command-line arguments** as follows:

```
leaderboardMaker sortingCriteria ChampionDataFile BattlesDataFile
```

- leaderboardMaker: When you run your program from the command line, you need start with its name. The name of the compiled program will be leaderboardMaker.
- sortingCriteria: This should be an integer value – 1 for actual win rate, 2 for expected win rate, and 3 for expectation skew.
- ChampionDataFile: This is the name of the file containing the names of the champions and their expected win rates.
- BattlesDataFile: This is the name of the file containing the battles' information

An example is provided below:

```
leaderboardMaker 1 champions.txt battles.txt
```

Internal Processing

As the program reads the information about each champion, it will create a corresponding data structure for each one as it reads the champion's name and their expected win rate, while initializing the number of battles, the number of wins, the actual win rate, and the expectation skew to zero at this stage. It will keep these structures in an array as illustrated below:

name: XQ3 expectedWinRate: 1 NumberOfBattles: 0 NumberOfWins: 0 ActualWinRate: 0 ExpectationSkew: 0	name: Ashya expectedWinRate: 0.25 NumberOfBattles: 0 NumberOfWins: 0 ActualWinRate: 0 ExpectationSkew: 0	name: Gingi expectedWinRate: 0.33 NumberOfBattles: 0 NumberOfWins: 0 ActualWinRate: 0 ExpectationSkew: 0	name: Cera2 expectedWinRate: 0.5 NumberOfBattles: 0 NumberOfWins: 0 ActualWinRate: 0 ExpectationSkew: 0	name: Crow expectedWinRate: 0.33 NumberOfBattles: 0 NumberOfWins: 0 ActualWinRate: 0 ExpectationSkew: 0	name: Alphi expectedWinRate: 0.75 NumberOfBattles: 0 NumberOfWins: 0 ActualWinRate: 0 ExpectationSkew: 0

The following functions that you will implement as part of this assignment are called in the following order in **the main function** for internal processing:

- **initializeChampions:** It takes the name of the data file containing the names of the champions and their expected win rates. It then reads the file and creates an array of data structure for the champions while initializing them as mentioned above. Finally, it returns the Champions array.
- **getBattleData:** It takes the Champions array, and the name of the data file containing the battles' information. It then computes the number of battles and wins for every involved champion as it reads through a battle.
- **computeWinRate:** It takes the Champions array, and then traverses the array for every champion and computes the actual win rate and the expectation skew as follows:

$$actual\ win\ rate = \frac{number\ of\ wins}{number\ of\ battles}$$

$$win\ rate\ ratio = \frac{actual\ win\ rate}{expected\ win\ rate}$$

$$expectation\ skew = |win\ rate\ ratio - 1|$$

- **heapSort:** It takes the Champions array and the sorting criteria which is either of the following three:
  1. actual win rate
  2. expected win rate
  3. expectation skew

  It firstly applies the Build Heap algorithm to create a max-heap and then sorts the data based on the heap sort algorithm.

Output

The program will print the sorted Champions leaderboard showing all the respective data for each champion (battles, wins, expected win rate, actual win rate, expectation skew). An example output is shown below where the sorting criteria is actual win rate.

```
Champion    Battles   Win       AWR       EWR       Skew
Ashya       4         3         0.75      0.25      2
Alphi       4         3         0.75      0.75      0
Crow        3         2         0.66      0.33      1
Cera2       2         1         0.5       0.5       0
XQ3         3         1         0.33      1         0.67
Gingi       3         0         0         0.33      1
```

Implement a function called **printLeaderboard** to display the data of the sorted array.

Notes

- The file containing the champions' names and expected win rates shall be named "champions.txt".
- The file containing the battles' data shall be named "battles.txt".
- Both files shall be stored in the same directory.
- The expected win rate CANNOT be equal to 0.
- Both win rates CANNOT be more than 1.

Incremental and Modular Development

You are expected to follow an incremental development approach. Each time you complete a function or module, you are expected to test it to make sure that it works properly. **You are also expected to follow modular programming which means that you are expected to divide your program into a set of meaningful functions.**

Professionalism and Ethics

You are expected to complete the assignments on your own. Sharing your work with others, uploading the assignment to online websites to seek solutions, and/or presenting someone else's work as your own work will be considered as cheating.

Rules

- You need to write your program by using C programming.
- You need to name your file with your student id, e.g. 1234567.c, and submit it to ODTUCLASS.
- Code quality, modularity, efficiency, maintainability, and appropriate comments will be part of the grading.
- **You need strictly follow the specifications given in this document.**

Grading Policy

The assignment will be graded as follows:

Grading Item	**Mark (out of 100)**
Champion Data Structure	3
Main function	16
initializeChampions	19
getBattleData	30
computeWinRate	9
heapSort	17
printLeaderboard	6