Project -3

Overview

The project is to create a turn-based role-playing game between two (computer) players in which they battle each other with a list of champions.

Gameplay

The game starts with 2 players having a list of random number of champions, as decided by the user. Each champion will be assigned a “role” and “level” based on some random probability. Each player has the same number of champions in the beginning. For both players, all the champions will always (at the start or any point in the gameplay) be sorted in the descending order of their “level”. In each turn, both the players will battle each other by deploying in their arena their best champion who is in the front of the list sorted according to the “level”. The outcome of the battle will be determined by the “role” and “level” of the two champions currently in the arena. After each turn, the current champion will be discarded from their list of champions and play the next champion. The battle can have different outcomes such as one or both players may gain extra (one or two) champions, or one or both players may lose extra (one or two) champions. Please see the table below on what the outcome is for a battle with two champions of the same or different “role”. The game continues until one player has lost all their champions. The player with any remaining champion wins.

Battle Outcome Possibilities

	MAGE (M)	**FIGHTER (F)**	**SUPPORT (S)**	**TANK (T)**
MAGE (M)	Player with the higher “level” wins. The winning player gains a new champion, and the losing player loses one champion. In a tie, nothing happens.	Player with the higher “level” wins. If MAGE wins, they gain a new champion, and the FIGHTER loses with no penalty. If FIGHTER wins, they gain no reward, but the MAGE loses one champion. In a tie, nothing happens.	Player with the higher “level” wins. If MAGE wins, they gain one new champion, and the SUPPORT loses two champions. Whereas, if SUPPORT wins, they get two new champions, and the MAGE loses one champion. In a tie, nothing happens.	TANK loses one champion. MAGE gains a new champion.
FIGHTER (F)		Both players gain a new champion.	Player with higher “level” wins. If SUPPORT wins, they gain one champion, and the FIGHTER loses but with no penalty. If	TANK player loses but with no penalty. FIGHTER player wins and gains a

			FIGHTER wins, they gain no reward, but the SUPPORT loses one champion. In a tie, nothing happens.	new champion.
SUPPORT (S)			Both players lose one champion.	TANK player wins and gets a new champion. SUPPORT player loses but with no penalty.
TANK (T)				Nothing happens – no penalty or reward.

Task 1 – create champion.h

- Create a **`struct`** named **`Champion`** that will hold information about a champion and will act as a node in a linked list. Each **`Champion`** will have the following information:
  - "role" of type **`ChampionRole`** (see **`enum`** below)
  - "level" of type integer
  - "next" champion for the linked list
- Create an **`enum`** for **`ChampionRole`** which can be one of the following four roles:

  `MAGE (M), FIGHTER (F), SUPPORT (S), TANK (T)`
- Declare all prototypes for the required functions (see Task 2) as well as the Header Guard.

Task 2 – create champion.c

Write the following functions which are typical linked list functions adapted for this game:

- **`Champion* createChampion()`** - This function dynamically allocates a new **`Champion`** struct object and returns a pointer to that struct object which will later be used to insert into a linked list.
  - The champion's "role" will be assigned randomly with a 25% probability each for **`MAGE (M), FIGHTER (F), SUPPORT (S), TANK (T)`**.
  - The champion's "level" will be assigned based on these random chances:
    - **`MAGE (M)`** - "level" is a random number between 5 and 8 inclusive.
    - **`FIGHTER (F)`** - "level" is a random number between 1 and 4 inclusive.

        - **`SUPPORT (S)`** - "level" is a random number between 3 and 6 inclusive.
        - **`TANK (T)`** - "level" is a random number between 1 and 8 inclusive.
- **`Champion* addChampion( Champion *head, Champion *c )`** - This function adds a new **`Champion`** struct object to the linked list that head is pointing at. It is assumed that a new **`Champion`** struct object is being passed into this function as parameter **c**. This function will add the new node in **descending (decreasing)** order of the champion's "level" value regardless of the "role".
- **`Champion* buildChampionList( int n )`** - This function builds a list of champions using a linked list. The parameter n determines how many champions are created. It will use **createChampion()** and **addChampion()** to create and return the head of the new linked list.
- **`void printChampionList( Champion *head )`** - This function traverses the linked list that head is pointing at and will print out the entire list of champions for a given player. Example: **`S6 M5 F4 S4 T3`**
- **`Champion* removeChampion( Champion *head )`** - This function removes and deallocates the first node in the linked list that head is pointing at. It returns the new head of the linked list.
- **`Champion* destroyChampionList( Champion *head )`** - This function is the destructor for a linked list that head is pointing at. It will remove all the champions from the player's list and return **NULL**.

TASK 3: Complete the `main.c`

- The program will accept one additional command line argument, which is the number of Champions each player will start with. Your program must determine if there is this command line argument:
    - If the command line argument does not exist, print an error, and end the program.
    - If the command line argument, check to see if it is a number greater than 0. If it is not, print an error and end the program.
- Build each player's Champion linked list based on the size passed in through the command line argument.
- Start the game loop. The game continues so long as both players have Champions to battle with. In each turn (iteration of the loop):
    - Print out the round number starting at 1.
    - Print out each player's Champions using **printChampionList()**.
    - Implement the outcomes for each permutation of Champion roles. There are 16 permutations of (MM, MF, MS, MT, FM, FF, FS, FT, SM, SF, SS, ST, TM, TF, TS, TT).
        - Using the table for battle outcomes, decide who wins / losses, and implement the reward / punishment on both players. For this, you will need to use the functions **createChampion()**, **addChampion()**, and **removeChampion()**.
        - Remember that at the end of each turn, the champions already deployed in the battle will be removed from the players' lists BEFORE implementing the reward / punishment based on the battle outcome.
    - Print the outcome for the battle and move to the next round.

- At the end of the loop, there will be either a tie (i.e., no player has any champions left) or one player who wins (i.e., still has champions). Determine and print this final outcome of the game.
- Remember to empty the entire list of champions for the players by freeing the memory for the linked list, before ending the program.

Task 4: Create a `makefile`

Create a **`makefile`** to compile and link all the files together. The grader will compile your code using your **`makefile`**. The name of the executable must be **`project3Exe`**

Submission

Be sure that your code follows the class coding style requirements. Your output should be similar in format as compared to the sample output shown below. Create a folder named your abc123, place all program files in this folder. Zip the folder and submit this abc123.zip file.

Rubric

[6 points] – General Requirements
    [2 points] – coding style – proper comments, indentation
    [2 points] – correctness – compiles properly and gives correct output
    [2 points] – submission – no missing files, zip, properly submitted

[2 points] – makefile
    [2 points] – make compiles and creates the executable with the correct name

[4 points] – champion.h
    [4 points] – header guard, enum, struct, function prototypes

[20 points] – champion.c file
    [4 points] – createChampion()
    [5 points] – addChampion()
    [3 points] – buildChampionList()
    [2 points] – printChampionList()
    [3 points] – removeChampion()
    [3 points] – destroyChampionList()

[18 points] – main.c
    [2 points] – command line argument validation
    [2 points] – build initial linked list for each player
    [2 points] – set up the game loop to continue playing until at least one player has Champions
    [2 points] – print out descriptive information for each round
    [8 points] – implement the functionality for each of the 16 permutations
    [2 points] – determine and print out the final outcome, free memory, and end program

SAMPLE OUTPUT

RUN-1

```
> ./project3Exe 5

============ PLAYER 1 V PLAYER 2 SHOWDOWN ============

----- ROUND 1 -----
Player 1: M5 F4 F2 F2 F2
Player 2: M6 T5 T3 F2 T1
Player 1 is a MAGE and Player 2 is a MAGE
Player 1 (MAGE) loses one champion.
Player 2 (MAGE) wins and gains one new champion.

----- ROUND 2 -----
Player 1: F2 F2 F2
Player 2: T5 T3 S3 F2 T1
Player 1 is a FIGHTER and Player 2 is a TANK
Player 1 (FIGHTER) wins and gains a new champion.
Player 2 (TANK) loses but with no penalty.

----- ROUND 3 -----
Player 1: F2 F2 T2
Player 2: T3 S3 F2 T1
Player 1 is a FIGHTER and Player 2 is a TANK
Player 1 (FIGHTER) wins and gains a new champion.
Player 2 (TANK) loses but with no penalty.

----- ROUND 4 -----
Player 1: F4 F2 T2
Player 2: S3 F2 T1
Player 1 is a FIGHTER and Player 2 is a SUPPORT
Player 1 (FIGHTER) wins but gains no reward.
Player 2 (SUPPORT) loses one champion.

----- ROUND 5 -----
Player 1: F2 T2
Player 2: T1
Player 1 is a FIGHTER and Player 2 is a TANK
Player 1 (FIGHTER) wins and gains a new champion.
Player 2 (TANK) loses but with no penalty.

============ GAME OVER  =============

Player 1 ending champion list: M7 T2
Player 2 ending champion list:
```

```
Player 2 ran out of champions. Player 1 wins.
```

RUN-2

```
> ./project3Exe 5

============ PLAYER 1 V PLAYER 2 SHOWDOWN ============

----- ROUND 1 -----
Player 1: M7 S6 S4 T3 F3
Player 2: T4 S4 T4 T3 F3
Player 1 is a MAGE and Player 2 is a TANK
Player 1 (MAGE) wins and gains a new champion.
Player 2 (TANK) loses the next champion.

----- ROUND 2 -----
Player 1: S6 S4 T3 F3 F3
Player 2: T4 T3 F3
Player 1 is a SUPPORT and Player 2 is a TANK
Player 1 (SUPPORT) loses but with no penalty.
Player 2 (TANK) wins and gains a new champion.

----- ROUND 3 -----
Player 1: S4 T3 F3 F3
Player 2: M8 T3 F3
Player 1 is a SUPPORT and Player 2 is a MAGE
Player 1 (SUPPORT) loses two champions.
Player 2 (MAGE) wins and gains one new champion.

----- ROUND 4 -----
Player 1: F3
Player 2: T3 F3 F1
Player 1 is a FIGHTER and Player 2 is a TANK
Player 1 (FIGHTER) wins and gains a new champion.
Player 2 (TANK) loses but with no penalty.

----- ROUND 5 -----
Player 1: T7
Player 2: F3 F1
Player 1 is a TANK and Player 2 is a FIGHTER
Player 1 (TANK) loses but with no penalty.
Player 2 (FIGHTER) wins and gains a new champion.

============ GAME OVER  =============

Player 1 ending champion list:
Player 2 ending champion list: T3 F1

Player 1 ran out of champions. Player 2 wins.
```

RUN-3

```
> ./project3Exe 5

============ PLAYER 1 V PLAYER 2 SHOWDOWN ============

----- ROUND 1 -----
Player 1: T6 F4 S4 S3 F2
Player 2: T8 T8 M6 T6 S4
Player 1 is a TANK and Player 2 is a TANK
Nothing happens - no penalty or reward.

----- ROUND 2 -----
Player 1: F4 S4 S3 F2
Player 2: T8 M6 T6 S4
Player 1 is a FIGHTER and Player 2 is a TANK
Player 1 (FIGHTER) wins and gains a new champion.
Player 2 (TANK) loses but with no penalty.

----- ROUND 3 -----
Player 1: S4 S3 F2 F1
Player 2: M6 T6 S4
Player 1 is a SUPPORT and Player 2 is a MAGE
Player 1 (SUPPORT) loses two champions.
Player 2 (MAGE) wins and gains one new champion.

----- ROUND 4 -----
Player 1: F1
Player 2: T6 S4 S3
Player 1 is a FIGHTER and Player 2 is a TANK
Player 1 (FIGHTER) wins and gains a new champion.
Player 2 (TANK) loses but with no penalty.

----- ROUND 5 -----
Player 1: S6
Player 2: S4 S3
Player 1 is a SUPPORT and Player 2 is a SUPPORT
Both players lose the next champion.

============ GAME OVER  =============

Player 1 ending champion list:
Player 2 ending champion list:

TIE -- both players ran out of champions.
```