Program 4 - Arrays, Structs, and Pointers

Due Date: 5:00 p.m., October 28 (Fri)

All programs will be tested on the machines in the Q22 lab. It is required that your code run on the computers in the lab (or via ssh).

Any changes made to the assignment after posting will be in red

Part 1: Tournament

In this lab you are going to implement your own tournament program with teams consisting of players. You will alter the driver code to give the teams whatever names you like.

- Part A
  - Create a Player struct that has 2 integer attributes:
    - offensive (int)
    - defensive (int)
  - Next you will need to create a struct called Team that contains a string buffer for the team name, and an array of up to 10 players.
  - Create a function:
    - Team * initializeTeam(int, char *)

    which takes a number of players on the team (int) and a team name (char *). The function should create a Team struct, and initialize all 10 players on the team. Assign random values between 1-10 for offense and 1-7 for defense for each player as they are created. The function should return a **pointer** to the newly created team (not a copy).
    - You may tweak the offensive and defensive number range however you like to give a more balanced result. These are the numbers that worked best for me.
    - The driver code will use your initializeTeam() function to create 8 teams, which will be placed into an array called league[], so make sure your function follows the expected interface.
  - Write another function:
    - Team * game(Team *, Team *)

    that takes pointers to two teams (Team *), then determines a winner using the team's total offensive and defensive score for each team's players plus some random element.

    Your game() function should:
    - The algorithm for determining the winner of a game is as follows:
      - Each team gets 10 attempts to score.
      - You must compare the defensive team's total defense with a random value between 0 and the offensive team's total offense.

            - If the final offensive value is greater than the defense, the team has a scored.
        - print out the team's names, scores, and the winner of the game
            - If the result is a tie, then you may determine the winner however you wish
        - Return a pointer to the winner.
    - Make sure this works correctly before moving on to the next part.
    - You will need to typedef your structs to remove the struct keyword in order to run the supplied driver code below.

- Part B
    - Once you have your game working and the result is random, create a function:
        - Team * tournament(Team **, int)

      that takes an array of pointers to Team structs, and the number of teams.
    - You must verify the number of teams is a power of 2. If it is not, print a message saying the number of teams is invalid and a NULL pointer.
    - Use your game function for each round to determine the rounds winners.
        - Because this is an elimination style tournament, each team should lose only once, while the winner goes on to the next round.
        - You will need to create unique matchups for each round between two teams, and discard the losers.
            - **MAKE SURE you do not delete the pointers from the league array. This will cause a memory leak.**
    - You will need to keep track of the winners each round, and match them up on the next round.
        - Do not assume you will only have 8 teams. Your code should work with any power of 2.
    - Once you have determined that the tournament() function works correctly, then go on to part C.

- Part C
    - Create a handicap mechanism that weighs the matches in favor of one team over another (your choice of how to implement).
    - Alter your game function to use the handicap to weight the tournament in favor of some teams, and run your results again.

Part 2 - Submission

- Required code organization:

    - program4.c //Driver Code
    - tournament.h
    - tournament.c
    - makefile
        - *You must have the following labels in your makefile:*
            - all - to compile all your code to an executable called '**program4**' (no extension). **Do not run**.
            - run - to compile if necessary and run
            - checkmem - to compile and run with valgrind
            - clean - to remove all executables and object files
- While inside your program 4 folder, create a zip archive with the following command
    - zip -r program4 *
        - This creates an archive of all file and folders in the current directory called program4.zip
        - **Do not zip the folder itself, only the files required for the lab**
- Upload the archive to Blackboard under Program 4.
- You may demo your lab by downloading your archive from Blackboard. Extract your archive, then run your code, show your source, and answer any questions the TA may have.

Grading Guidelines

Total: 20 Points

- **Part A (8 points):**
    - Passes tests 1-4 (1 point each)
    - Players and Team structs are initialized as described (2 points)
    - Game function implemented as described: (2 points)
- **Part B (10 points):**
    - Tournament function checks that number of teams is a power of 2 (2 points)
    - Tournament function implemented as described: (3 points)
    - Passes tests 5-7 (1 point each)
    - Passes test 8 (2 points)
- **Part C (2 points):**
    - Handicap is implemented: (2 points)
- **Style Guidelines and Memory Leaks**
    - You will lose significant points for the following:
        - Makefile does not have requested format and labels (-10 points)
        - Does not pass Valgrind Tests (-10 points)
        - Does not follow requested program structure and submission format (-10 points)
        - Does not follow formatting guidelines (-5 points)