# Program 4 - Files and Structures

CS 580U - Fall 2017

Due Date: 5:00 p.m., November 10, 2017

*Any changes/updates to the project description will be in red -* ***Last Updated 11/06 13:17 p.m.***

*All programs will be tested on the machines in the Q22 lab. If your code does not run on the system in this lab, it is considered non-functioning EVEN IF IT RUNS ON YOUR PERSONAL COMPUTER. You can write your code anywhere, but always check that your code runs on the lab machines before submitting.*

## Driver Code and Test Input Files

- **Provided Files**
  - program4.c //Driver Code
  - players.dat

## Grading Rubric

***TOTAL: 25 points***

- **Part A (8 points):**
  - Test 1: Reads in player file correctly (4 points)
  - Test 1: Initializes Teams correctly (4 points)
- **Game function works as described are initialized as described (8 points)**
  - Test 2: Sanity check for null pointer (2 points)
  - Test 3: Game function implemented as described: (4 points)
  - Test 4: Team can play against itself (2 points))
- **Part B (9 points):**
  - Test 5: Ensure teams are a power of two (1 point)
  - Test 6: Tournament results in a single winner (3 points)
  - Test 7: Tournament results in a random winner (2 points)
  - Test 8: Cleans up memory for each Team(3 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 (-5 points)
    - Does not follow requested program structure and submission format (-10 points)

## Guidelines

This is an individual assignment. You must do the vast majority of the work on your own. It is permissible to consult with classmates to ask general questions about the assignment, to help discover and fix specific bugs, and to talk about high level approaches in general terms. It is not permissible to give or receive answers or solution details from fellow students.

You may research online for additional resources; however, you may not use code that was written specifically *to* solve the problem you have been given, and you may not have anyone else help you write the code or solve the problem. You may use code snippets found online, providing that they are appropriately and clearly cited, within your submitted code.

*By submitting this assignment, you agree that you have followed the above guidelines regarding collaboration and research.*

# 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, but you will read the players' data from a file.

- ## Part A
    - Create a Player struct that has the following attributes:
        - offensive (int)
        - defensive (int)
        - number (int)
        - team (int)
        - first (char *)
        - last (char *)
    - Next you will need to create a struct called Team that contains:
        - a string buffer for the team name (char *)
        - a pointer to an array of players (Player *)
            - You may add any additional attributes you require
    - Create the following functions:
        - Player * draftPlayers(char * filename, int team, int num_players)
            - The draft players function takes a filename for a file containing players in the following format:
              *<team #>, <first name>, <last name>, <player_num>, <offense>, <defense>*
                - Each player information will be on a separate line
                - You are guaranteed to only have well formed files.
            - The function should return an array of <num_players> for the given team number
        - Team * initializeTeam(char * name, Player * players)
            - which takes a team name (char *) and an array of players.
            - The function should:
                - Create a Team struct,
                - The function should return a **pointer** to the newly created team (not a copy).
        - The driver code will use your initializeTeam() function to create 8, 16, or 32 teams, which will be placed into an array called league[], so make sure your function follows the expected interface.
    - Next you will write the function:
        - Team * game(Team *, Team *)

      that takes pointers to two teams (Team *). Your game() function should complete the following:
        - 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 players 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.
        - Return a pointer to the winner.
    - Make sure this works correctly before moving on to the next part.
    - *NOTE: 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 (8 | 16 | 32).
    - Lastly, you will need to write a function that cleans up memory for each team:
        - `void deleteTeam(Team *);`

# Part 2 - Submission

- Required code organization:
    - program4.c //Driver Code
    - tournament.h
    - tournament.c
    - players.dat
    - 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 <yourid>_program4 <files to include>
        - 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.