ESC190H1S Winter 2022: Lab 4

TA evaluations in practicals: March 22nd/23rd, 2022
Code submission due: March 24rd, 2022 at 11:59pm on Gradescope

Overview

The estimated completion time for this lab is 3 hours. Starter code is provided for this lab on Quercus. You will submit **only** `lab4.c`. Please carefully review the provided starter code entirely before starting to write your own code.

Your code is expected to work when compiled with the command

```
gcc -o <target executable> tournament.c lab4.c
```

Starter tests are provided on Gradescope, but the provided Gradescope feedback is *intentionally* vague to urge you to test your own code. Please note, this lab has hidden test cases that are not visible to you on Gradescope that will be used for grading. Make sure to thoroughly test the functionality and memory management of your code yourself.

Additional Requirements

- You **MUST NOT** add any `#include` statements, or modify the ones given.
- You **MUST NOT** add a `main` function to `lab4.c`.
- Your code **MUST** work if any of the `#define` directives in `lab4.h` are modified.
- Do **NOT** add any additional print statements to your code other than what is indicated in the lab handout. Do not read from any filestreams, and do not write to any filestreams other than `stdout`.

Violating any of these constraints means that we can not guarantee your code will function when grading and could result in a grade of zero.

Background

Scuffle Games, a trendy video game company with a proven track record of success with games like Legends of Lore, has offered you an exciting opportunity: an unpaid internship as a software developer! Lured in by the opportunity to have a "good learning experience" and "projects to add to your resume", you decide to take the unpaid internship, despite your concerns that the work you will be doing is highly skilled work that the company will end up massively profiting from.

Scuffle Games has a new 1-versus-1 first-person shooter game in beta testing. Scuffle Games intends to introduce the game to the E-sports scene, but midway through beta testing, the testers pointed out that the developers forgot to build record-keeping for the tournaments into the game! Due to a tight production schedule and limited staff, upper management has denied the development team's request to adjust the release timeline to double back and integrate the record-keeping into the game itself. Your manager, clearly stressed by this, asks you to create a separate software for the record-keeping and ignore the issue of integrating it with the game for now. You're not sure this is a good idea, but being a new "employee", you don't feel like you can really comment.

Their proposed tournament structure is described as follows: each *match* consists of two players. The two players compete in rounds, with the winner of the *match* being the first player that wins 13 rounds. There are no draws possible in the game. The winner of the match proceeds to the next *bracket*. The *bracket*s are used to determine player *rank*. Once a player wins a match, they advance to the next bracket to compete for the next rank. If the player loses a match, they are eliminated, and do not play in any further matches.[1]

Tournament Tree Structure

Carefully review the declared `struct`s in `lab4.h`. You may also find it helpful to review `tournament.c` for example usage.

The tournament tree is built in units of `PlayerRecord`s. A `PlayerRecord` keeps track of an associated `Player` using the `player` field. Each match is represented as a pair of `PlayerRecord`s. The `parent` field of a `PlayerRecord` indicates the winner of the match. The `left_child` and `right_child` fields of the `PlayerRecord` indicate the loser and winner, respectively, of the match from the previous bracket.

The `game_records` field is a static, two-element integer array that stores the number of rounds the `Player` won at index `WINS` and the number of rounds the `Player` lost at index `LOSSES` (see `lab4.h` for these constants).

[1] This lab was inspired by Valorant and Haikyuu.

Tasks

Complete the following functions in `lab4.c`. The function descriptions are provided in `lab4.c`.

- `Player* create_player(...)`
- `PlayerRecord* create_leaf_record(...)`
- `PlayerRecord* add_match(...)`
- `int get_player_rank(...)`

Your code is expected to run Valgrind clean when interfaced with the provided `clear_tournament_records(...)` function (you can see example usage in `tournament.c`).

Consider the provided code for `clear_tournament_records(...)` and `print_with_rank(...)`.

What are the best- and worst- case runtimes of the provided implementations?

Identify any modifications you can make to the implementations to improve their runtimes, if any. Explain to your TA.

Definitions

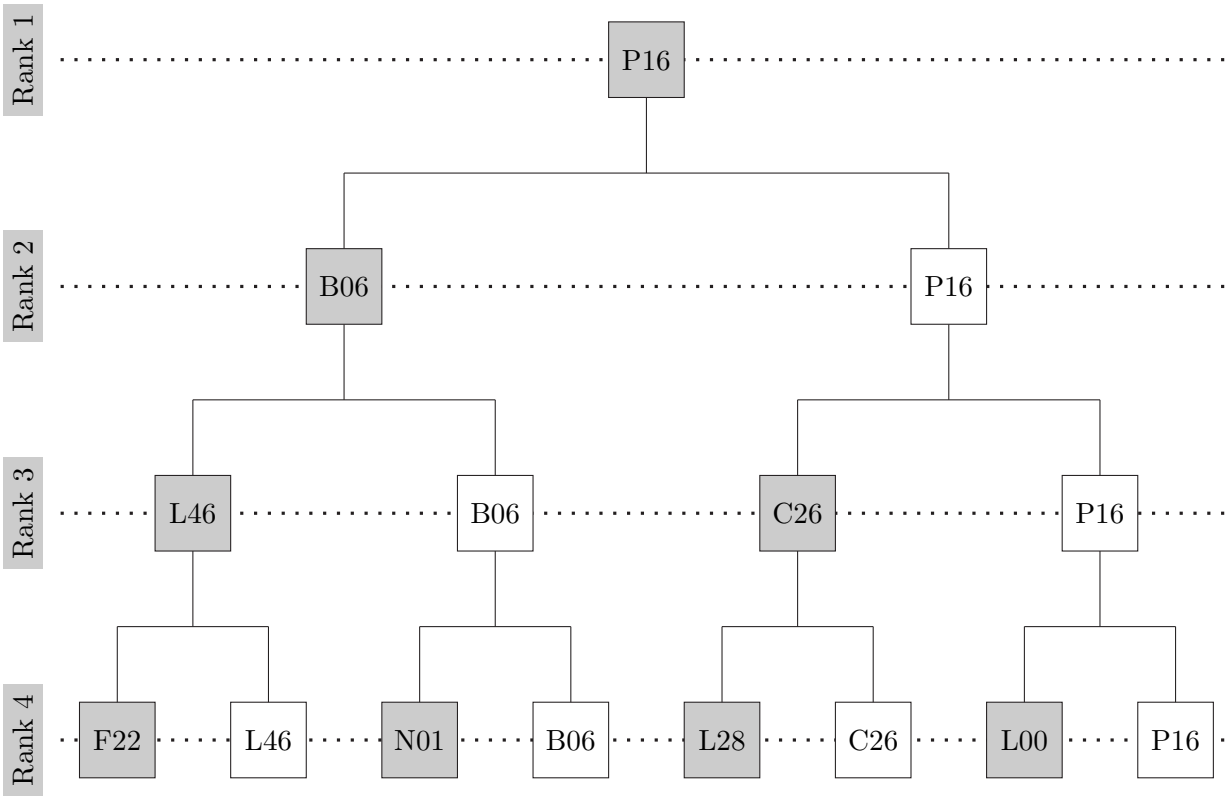
Leaf record: a record from the first bracket of the tournament. There is one for each player in the tournament. We guarantee that there are an even number of players at the first bracket.

Completed tournament: a tournament that has concluded; i.e., the finals have taken place and a rank 1 winner has been determined.

Player rank: player rank can only be determined for a completed tournament. The winner of the finals bracket is awarded rank 1. For any other player, let `n` be the *depth* of the PlayerRecord where that Player loses a match. Then, the player's rank is `n + 1`.

Sample Tournament

The completed tournament from `tournament.c` is depicted below.



A written description of the tournament is provided on the following page.

First bracket: 4 separate matches across 8 players.

N01 vs B06: 10 v 13, B06 won (Round 1 Outcome 1)
F22 vs L46: 5 v 13, L46 won (Round 1 Outcome 2)
L28 vs C26: 2 v 13, C26 won (Round 1 Outcome 3)
P16 vs L00: 13 v 0, P16 won (Round 1 Outcome 4)

Second bracket: 2 matches between the winners of the first bracket.

B06 vs L46: 13 v 7, B06 won (Round 2 Outcome 1)
C26 vs P16: 10 v 13, P16 won (Round 2 Outcome 1)

Finals bracket: 1 match between the winners of the semifinals.

B06 vs P16: 8 v 13, P16 won (Round 3 Outcome)

Ranks: P16 is rank 1; B06 is rank 2; L46 and C26 are rank 3; F22, N01, L28, and L00 are rank 4.