CSCI 151: Programming for Scientists and Engineers

Programming Assignment 4

Deadline: Friday 20 November 2020 at 20:00

This programming assignment is to assess your ability to develop a reasonably sized program. The programming assignment is based on the contents of Lessons 1–28 and **will be mostly graded in an automated way.** You will need to submit on Moodle two files as explained in the rules below.

Below you find

- the rules that you have to adhere in carrying out and submitting the assignment (please read them carefully);
- four mandatory tasks and one optional task;
- important notes (see the final page).

Please work at the assignment at the same pace as you study, according to the schedule on the course website. The **only three libraries you can use** are: `stdio.h`, `stdlib.h` and `string.h`. You **must** also follow the **Important Notes** at the end of this assignment.

Rules

- Late submissions will NOT be accepted. A 0 grade will be given for late submissions.
- No deadline extensions will be granted.
- The mandatory part of this assignment (**Tasks 1–4**) is graded according to the following **rubric**:

  1 point for correct indentation;

  1 point for appropriate comments;

  4 points for each task, of which **1 point** is for the appropriate testing of that specific task.

  If the code of a task does not compile, you will receive 2 points penalty and the task will be graded manually.
- **Your code must be done individually. You cannot collaborate with your peers. You cannot post your code on any public repository, neither can you share it with anybody.**
- Check your code with the online compiler `https://repl.it/languages/c` before submitting it to make sure it compiles. However, even if you get the expected results from the online compiler, this does not guarantee that your code is correct (you can only claim that your program compiles, not that it is correct).
- Submitted code will be automatically checked using tools that detect plagiarism.
- In case **cheating** is detected, **the grade of this assignment will be zero** and **the grades of all your previous assignment will be changed to zero** (in total **you would lose 40% of your final grade!**).
- You may be asked to explain your code and rewrite some parts of it in front of the instructors.
- Submit all tasks within the same `.c` file named as `YourlastnameYourfirstname.c` **without creating any archive**.
- **In order to be accepted, your submission**
  - **must include two files** (**no Word or other formats and no archives!**):
    1. one single file in text format with a '`.c`' extension, correctly named as explained above, **and**
    2. the text file you used for testing Tasks 3 and 5 (you can choose any name for this testing file);
  - **The C file must include the string assignment with your first name, last name and student number as explained in items 1 and of the Important Notes at the end of this assignment**.

  You will receive a zero grade if you do not submit the assignment or you do not adhere to this rule!

Problem

You are going to write a program to manage football tournaments by defining two structures to store information about teams and matches and by reading data from text files in order to

(Task 1) initialise an array of structures that stores team information;

(Task 2) use the match results to update the team information on points and goals;

(Task 3) print the tournament standings to a text file and to the console output;

(Task 4) store a match result in a structure;

(Optional Task 5) print an ordered version of the tournament standings.

Tasks

Task 1 [4 points] Using the `typedef` construct, define a `struct` called `Team` to describe a football team, with the following fields:

- a string `name` of 20 characters, inclusive the `NULL` character, to store the name of the team (one word with no spaces);
- a string `city` of 20 characters, inclusive the `NULL` character, to store the city of the team (one word with no spaces);
- an integer `points` giving the number of points collected by the team during the football tournament;
- an integer `goalFor` giving the number of goals scored by the team during the football tournament;
- an integer `goalAgainst` giving the number of goals scored against the team during the football tournament;

Define the function

```
int initTeams(Team teams[32])
```

which

- reads a text file `teams.txt` with the same structure as the following

```
Ordabasy    Shymkent
Tobol   Kostanay
Zhetysu Taldykorgan
Astana Nur-Sultan
Atyray Atyrau
Kairat Almaty
```

```
Okzhetpes Kokshetau
Kaisar Kyzylorda
Aktobe Aktobe
Irtysh Pavlodar
Shakhter Karagandy
Taraz Taraz
```

and stores the name and the city of each team in array `teams`, while initialising fields `points`, `goalFor` and `goalAgainst` to 0;

- returns the number of entries stored in array `teams`.

Write in the `main` function some code to appropriately test the `initTeams` function.

Task 2 [**4 points**] Define the function

```
int addResults(int n, Team teams[32])
```

which

- reads a text file `match-results.txt` with the same structure as the following

```
7 Astana Nur-Sultan 3 0 Atyrau Atyrau
8 Irtysh Pavlodar 0 1 Astana    Nur-Sultan
   7 Kairat Almaty 2 1 Tobol   Kostanay
7 Kaisar Kyzylorda 1 1 Aktobe Aktobe
```

- updates the array `teams` of `n` teams (stored in the first `n` positions of the array) by modifying the information associated with each team involved in the matches described in file `match-results.txt` by
  - adding
    - either 3 points to the winner,
    - or 1 point to both teams in case of a draw,
  - updating the goal scored and against;
- returns the number of matches processed, taking into account that only matches with teams stored in `match-results.txt` are processed.

Note that each line in file `match-results.txt` includes from left to right the following elements: the match day, the name of the host team, the city of the host team, the goals scored by the host team, the goals scored by the guest team, the name of the guest team and the city of the guest team. There might be any number of spaces and/or tabs between each pair of elements as well as at the beginning and the end of the line.

Write in the `main` function some code to appropriately test the `addResults` function. You may just use file `match-results.txt` to test the function without creating any

additional testing file.

Task 3 [**4 points**] Define the function

```
Team* printStandings(int n, Team teams[32], char fileName[20])
```

which

- writes the table of the standings of the `n` teams stored in array `teams` to a text file `fileName` and to the console output with the same structure as the following

  ```
  TEAM              P    GF    GA     GD
  Astana           21    31     7    +24
  Kairat           18    25     9    +16
  Tobol            18    26    13    +13
  Atyrau           17    24    10    +14
  ...
  Okzhetpes         7    11    20     -9
  Aktobe            5    12    18     -6
  ```

  where column `P` shows the points, column `GF` shows the goals scored, column `GA` shows the goals against and column `GD` shows the goal difference. Moreover, the function returns
  - a pointer to the champion team, if there is a single champion team, that is, the team
    * with the highest number of points, or
    * the team with the highest number of points and the highest goal difference, if there are more teams with the highest number of points, or
    * the team with the highest number of points, the highest goal difference and the highest number of scored goals, if there are more teams with the highest number of points and the highest goal difference;
- a `NULL` pointer, if there are more champion teams.

Note that the table does not need to be ordered by number of points (although we have ordered it in the example). Moreover, you can can freely choose the number of characters for the width of each column, provided that the names of the teams are aligned to the left (first column) and the numbers are aligned to the right (other columns).

Hint: you can print an integer with the sign `+` or `-` as requested in the last column of the table using the formatting string `"%+i"` (or `"%+`m`i"`, if you need the add enough spaces to the left to fill in m characters).

Write in the `main` function some code to appropriately test the `printStandings` function. In order to appropriately test the function, you will need to create a new

text file that covers the four requirements considered in the function specification. **Do not forget to submit this new text file!**

Please make sure to include comments that explain how your testing code covers all four requirements.

Task 4 [4 points] Using the `typedef` construct, define a `struct` called `Match` to describe a football match between two teams, with the following fields:

- an integer `matchDay` giving the number of the match day;
- a pointer `*host` to the host team of type pointer to `Team`;
- a pointer `*guest` to the guest team of type pointer to `Team`;
- an integer `hostScore` giving the score of the host team;
- an integer `guestScore` giving the score of the guest team.

Define the function

```
Match* storeResult(Team *host, Team *guest)
```

which

1. reads a text file `match-results.txt` searching for the match between `host` and `guest` and returns a `NULL` pointer if the match is not found;
2. if the match is found in file `match-results.txt`, the function
   - dynamically allocates space for a new element of type `Match`;
   - assigns the values found in `match-results.txt` to the newly allocated element of type `Match`;
   - returns
     - a pointer to the newly created element of type `Match`;
     - a `NULL` pointer, if no match is found or no memory can be allocated.

You can assume that `match-results.txt` contains at most one match between the two teams.

Write in the `main` function some code to appropriately test the `storeResult` function. You may just use file `match-results.txt` to test the function without creating any additional testing file. Although you need to make sure that the function returns a `NULL` pointer in the case specified above, you do not need to submit any testing code or additional text file for this specific case.

Optional Task 5 [**4 bonus points**] Define the function

```
Team* printOrderedStandings(int n, Team teams[32], char fileName[20])
```

which is a new version of function `printStandings` from Task 3 that orders the table in decreasing order according to

1. the number of points (**1 bonus point**);
2. the goal difference, if the number of points is the same (**1 bonus point**);
3. the number of scored goals, if the number of points and the goal difference are the same (**1 bonus point**);
4. the team name alphabetic order, if the number of points, the goal difference and the scored goals are the same (**1 bonus point**);

Write in the `main` function some code to appropriately test the `printOrderedStandings` function. You may use the same file you used for testing Task 2.

This optional task will be grade only if

- you get at least 2 points for each of the four mandatory tasks;
- you get at least 12 points in the mandatory part of the assignment;
- your code compiles.

The grading of this task will be entirely automatic. Each bonus point will be given only if your code contains the appropriate tests for the corresponding requirement. For example the bonus points for requirement 1 will be given only if you test the ordering according to the number of points.

Bonus points will be added directly to the course final grade.

Please note that if cheating is detected in this task all rules about cheating will be applied to the entire assignment.

Important Notes

1. **Your program must have the structure**

```
⟨Inclusion of libraries and definition of structures⟩
⟨The required functions and possible additional functions⟩
int main(void) {
   setvbuf(stdout, NULL, _IONBF, 0);
   char firstName[] = "⟨your first name⟩";
   char lastName[] = "⟨your last name⟩";
   char studentID[] = "⟨your student number⟩";
   printf("Student:  %s %s\n\n", firstName, lastName);
   ⟨Testing of Task 1⟩
```

⟨*Testing of Task 2*⟩
⟨*Testing of Task 3*⟩
⟨*Testing of Task 4*⟩
⟨*Testing of the Optional Task 5 (if done)*⟩
`return 0;`

You must assign your first name, last name and student number to the variables above to allow for the automatic grading.

2. **You must use** the **type names** (e.g. `Team` and `Match`), **all their fields names** and the **function names** indicated in the text of the assignment. Your program will undergo an automatic testing and any changes of such names will result in a test failure and, as a consequence, in a decrease of your grade.

3. Feel free, instead, to change the names of the formal parameters of functions (**but not their types and orders!**) and to make use of additional functions naming them as you like.

4. **The functions requested in the assignment must behave exactly as specified** in the text of this assignment and **be as general as requested**. Be aware that

   - **any form of hard coding**,
   - **cumbersome code**,
   - **any unnecessary code**, or
   - **any code that does extra things that are not explicitly requested in the specification (such as input-out interactions or printouts)**

   will result in point deductions or even in 0 points for the task.

5. You should make an effort to submit code that compiles to avoid to lose points. However, it is still better to submit code that does not compile but show that you tried all tasks, rather than getting stuck on a task and not having time to complete the others;

6. Finally, it is better if you do not submit the assignment, rather than copying it: if you do not submit you would lose 18% of your final grade, whereas, in case of academic misconduct (copying code from peers or from any public sources, giving code away to peers, sharing code or publishing code, getting or purchasing the code from a third party), **you would lose 40% of your final grade** and your action would have **further serious consequences**: academic misconduct reported to the school and failing of the course in case of repeated misconduct.