Structuri de Date
Tema 1: Echipe de fotbal

Alexandra Ștefania Ghiță

18 martie 2019

George este pasionat de fotbal și vrea să își alcătuiască propria echipă de fotbal cu care să câștige cât mai multe trofee. El are acces la listele de jucători ale unor cluburi internaționale, pe care vrea să le analizeze o perioadă de timp ca să își poată da seama care sunt cei mai promițători jucători.

Scopul temei este să îl ajutăm pe George să țină evidența jucătorilor în mod eficient ca să își formeze echipa cea mai bună.

Descrierea problemei:

Tema va fi implementată în limbajul C, folosind liste înlănțuite. Pentru reprezentarea jucătorilor veți folosi o structură de listă dublu înlănțuită, pentru a facilitata adăugarea și ștergerea de elemente. Pentru reprezentarea echipelor veți folosi o listă simplu înlănțuită, fiind necesară doar adăugarea de elemente la finalul acesteia.

Fiecare jucător este caracterizat prin nume, poziție de joc, scor și un indicator care precizează dacă jucătorul este accidentat sau nu. Scorul cuantifică abilitățile unui jucător și se poate modifica după fiecare meci în care acesta a jucat. Aceasta este reprezentat de un număr întreg, -100 <= scor <= 100. Se garantează că jucătorii au nume unice.

Fiecare club are un nume și două liste de jucători, una cu jucătorii neaccidentați (cei care pot juca) și una cu jucătorii accidentați (cei care nu pot juca). Listele vor fi păstrate în ordine, după cum urmează:

- Lista cu jucătorii accidentați este ordonată alfabetic după nume
- Lista de jucători neaccidentați este ordonată după următoarele criterii, în ordine:
    - alfabetic după pozițiile de joc (pozițiile de joc posibile așa cum apar în teste sunt: *atacant*, *fundas*, *mijlocas*, *portar*)
    - descrescător după scor
    - alfabetic după nume

Se garantează că numele cluburilor sportive sunt unice.

Numărul de echipe este variabil în timp, noi cluburi putând fi adăugate oricând. Există posibilitatea ca la un moment dat unele cluburi să nu poată participa la competiții, având mai puțin de 11 jucători.

Când un jucător se accidentează, acesta este mutat în lista de jucători accidentați, iar scorul acestuia este modificat proporțional cu numărul de zile pe care le stă pe bancă (scorul va fi trunchiat la valoarea întreagă):

```
scor_nou = scor_vechi - 0.1 * nr_zile_accidentare
```

Cerințe:

A. Țineți evidența jucătorilor de fotbal, implementând următoarele funcții:

- **`initilize_clubs(n, names)`**
  - inițializează o listă de cluburi sportive. *n* reprezintă numărul de cluburi, iar *names* este un vector care conține numele cluburilor, în ordinea în care se vor adăuga.
- **`add_club(name)`**
  - adaugă clubul sportiv cu numele *name*. Adăgarea se va face la sfârșitul listei de cluburi.
- **`add_player(club_name, player_name, position, score)`**
  - adaugă jucătorul *player_name* în clubul *club_name* în lista asociată jucătorilor neaccidentați. Adăgarea se va face ținând cont de ordinea prezentată anterior.
- **`transfer_player(player_name, old_club, new_club)`**
  - mută jucătorul *player_name* din clubul *old_club* în clubul *new_club*, dacă este posibil. Toate caracteristicile jucătorului vor rămâne nemodificate în urma transferului. Listele vor rămâne ordonate corect în urma operației.
- **`remove_player(club_name, player_name)`**
  - elimină jucătorul *player_name* din clubul sportiv *club_name*, dacă există.
- **`update_score(club_name, player_name, score)`**
  - actualizează scorul unui jucător la valoarea *score*. Listele vor rămâne ordonate corect în urma operației.
- **`update_game_position(club_name, player_name, position, score)`**
  - modifică poziția de joc a unui jucător și îi atribuie un nou scor pentru aceasta. Listele vor rămâne ordonate corect în urma operației.
- **`add_injury(club_name, player_name, days_no)`**
  - semnalează o accidentare a jucătorului *player_name* care nu va putea juca *days_no* zile. Jucătorul va fi mutat în lista jucătorilor accidentați. Listele vor rămâne ordonate corect în urma operației.
- **`recover_from_injury(club_name, player_name)`**
  - anunță recuperarea jucătorului *player_name*. Acesta va fi mutat în lista jucătorilor neaccidentați. Listele vor rămâne ordonate corect în urma operației.

B. Extrageți echipe posibile, în funcție de anumite criterii de căutare. Fiecare funcție va întoarce o listă cu elemente nou create:

- **`union_teams(club_A, club_B)`**

    - întoarce lista cu toți jucătorii neaccidentați ai cluburilor sportive *club_A* și *club_B*. Lista întoarsă va păstra proprietatea de ordonare a listelor de jucători neaccidentați, prezentată anterior.
- **`get_best_player(position)`**
    - întoarce cel mai bun jucător neaccidentat ca scor pentru poziția de joc *position*. În cazul în care sunt mai mulți jucători cu același scor, se va alege primul jucător în ordine alfabetică.
- **`get_top_players(N)`**
    - întoarce cei mai buni N jucători neaccidentați ca scor din fiecare club sportiv. Lista va fi ordonată descrescător după scor, apoi crescător după nume. În cazul în care există mai puțini jucători decât numărul cerut, atunci vor fi adăugați doar aceia. În cazul în care sunt mai mulți cu același scor, se vor alege primii în ordine alfabetică.
- **`get_players_by_score(score)`**
    - întoarce toți jucătorii cu scorul cel puțin egal cu *score*. Lista va fi ordonată descrescător după scor, apoi crescător după nume.
- **`get_players_by_position(position)`**
    - întoarce toți jucătorii care joacă pe poziția *position*. Lista va fi ordonată descrescător după scor, apoi crescător după nume.
- **`get_best_team()`**
    - întoarce cea mai bună echipă, cu jucătorii cu cele mai mari scoruri pentru următoarele poziții de joc: 1x portar, 4x fundaș, 3x mijlocaș și 3x atacant. Lista va fi ordonată descrescător după scor, apoi crescător după nume. În cazul în care există mai puțini jucători decât numărul cerut, atunci vor fi adăugați doar aceia. În cazul în care sunt mai mulți jucători cu același scor se vor alege primii în ordine alfabetică.

Implementare și notare:

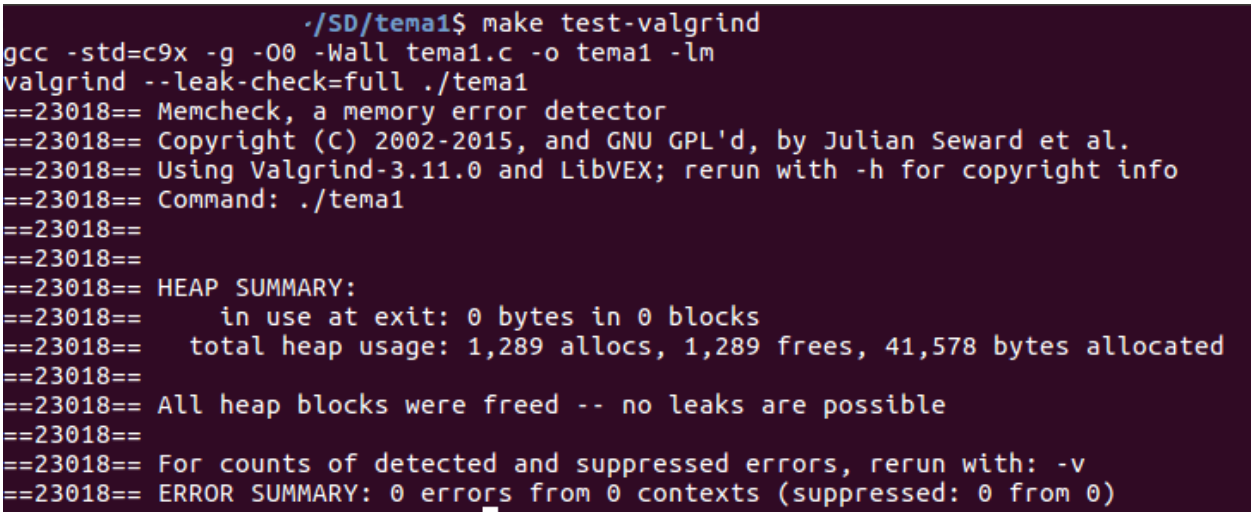
Pentru implementarea temei este pus la dispoziție un schelet de cod pe care îl veți completa cu funcționalitățile cerute. Pentru cerința 1 veți completa fișierul *FootballClub.h*, iar pentru cerința 2 *TeamExtractor.h*. Nu aveți voie să modificați structurile definite în fișierul *FootballClub.h*.

Punctajul temei este de 100 de puncte împărțite astfel:

- 85 puncte: teste. Pentru testarea temei rulați scriptul *run.sh* (./run.sh). Acesta va rula fișierul tema1.c și va compara rezultatul obținut cu cel de referință.

- 5 puncte: coding style - codul trebuie să fie comentat, consistent și ușor de citit. Tema nu trebuie să conțină: warninguri la compilare, linii mai lungi de 80 de caractere, tab-uri amestecate cu spații, o denumire neadecvată a funcțiilor sau a variabilelor, folosirea incorectă de pointeri, neverificarea codurilor de eroare, utilizarea unor metode ce consumă resurse în mod inutil, neeliberarea resurselor folosite sau alte situații nespecificate aici, dar considerate inadecvate.

- 5 puncte: eliberarea memoriei. În urma execuției programului toate resursele trebuie să fie eliberate și memoria să fie accesată în mod corect (fără erori în valgrind). Pentru a verifica eliberarea memoriei puteți folosi target-ul *test-valgrind* din fișierul Makefile, care se folosește de utilitarul valgrind. (make test-valgrind). Pentru a elibera resursele trebuie să completați funcțiile *destroy_club_list* și *destroy_player_list* din fișierul *FootballClub.h.* Dacă lucrul cu memoria și eliberarea resurselor a fost făcută în mod corespunzător, atunci rezultatul utilitarului valgrind va arăta similar cu următoarea imagine:



  În urma rulării s-au obținut 0 erori (ERROR SUMMARY) și toate blocurile au fost eliberate (HEAP SUMMARY).

- 5 puncte: README – va conține detalii despre implementarea temei.

Temele care nu compilează, nu rulează sau obțin punctaj 0 la teste, indiferent de motive, vor primi punctaj 0.