Structuri de Date

2021-2022

Tema 1: Liste

14.03 - 03.04 (deadline hard)

Responsabil: Alexandra Ștefania Ghiță

Obiectiv: Obiectivul acestei teme este înțelegerea și aprofundarea listelor și operațiilor pe liste.

Enunț: Gara din Hogsmeade se dezvoltă rapid și are nevoie de o digitalizare a informației pentru a-și planifica mai ușor activitățile. Scopul nostru este să creăm un program care să rezolve operațiile cele mai importante ale gării, pentru a ușura munca angajaților.

Pentru a putea implementa funcțiile necesare, vom folosi trei structuri de date:

a). *TrainStation* este structura care reprezintă gara. Gara are mai multe peroane, iar pe fiecare peron poate staționa un tren. Câmpul *platforms_no* reprezintă numărul de peroane al gării, iar câmpul *platforms* este un vector care conține peroanele și trenurile staționate. Nu este obligatoriu ca pe fiecare peron să existe un tren staționat la un anumit moment de timp.

```
struct {
      int platforms_no;
      Train **platforms;
} TrainStation;
```

b). *Train* este structura care reprezintă un tren. Fiecare tren este format dintr-o locomotivă și o secvență de vagoane. Câmpul *locomotive_power* reprezintă greutatea maximă pe care o poate transporta locomotiva. Dacă greutatea totală a vagoanelor depășește această valoare, trenul nu va putea pleca din stație. Câmpul *train_cars* reprezintă primul vagon din secvența de vagoane a trenului (este vagonul atașat de locomotivă).

```
struct {
      int locomotive_power;
      TrainCar* train_cars;
} Train;
```

c). *TrainCar* este structura care reprezintă un vagon. Fiecare vagon are o greutate asociată care este reținută în câmpul *weight*. Câmpul *next* reprezintă următorul vagon din secvența de vagoane a trenului.

```
struct {
      int weight;
      TrainCar *next;
} TrainCar;
```

Vom implementa 3 tipuri de funcții pentru gara din Hogsmeade:

- **funcții de operare** - sunt funcții care modifică structura trenurilor de pe peroane. Putem să adăugăm sau să scoatem un tren de pe un peron, să adăugăm vagoane sau să scoatem din vagoanele unui tren.
- **funcții de căutare** - sunt funcții care ajută la găsirea trenurilor cu anumite caracteristici. Putem căuta trenul cel mai rapid, cel mai bine încărcat, încărcat necorespunzător sau cu o distribuție slabă a greutății.
- **funcții de organizare** - sunt funcții care ajută la organizarea mai bună a greutății trenurilor. Putem să reordonăm vagoanele unui tren sau să eliminăm un vagon dintr-un tren încărcat necorespunzător.

Cerința 1 (30p). Implementați următoarele funcții de operare a trenurilor:

a). **`arrive_train(station, platform, locomotive_power)`** - adaugă o locomotivă cu puterea *locomotive_power* pe peronul cu numărul *platform*. (5p)

b). **`leave_train(station, platform)`** - eliberează peronul cu numărul *platform*. (5p)

c). **`add_train_car(station, platform, weight)`** - adaugă un vagon cu greutatea *weight* la trenul de pe peronul *platform*. Vagonul va fi adăugat la capătul trenului. (5p)

d). **`remove_train_cars(station, platform, weight)`** - scoate toate vagoanele care au greutatea *weight* din secvența de vagoane a trenului de pe peronul *platform*. (5p)

e). **`move_train_cars(station, platform_a, pos_a, cars_no, platform_b, pos_b)`** - mută *cars_no* vagoane începând cu vagonul aflat pe poziția *pos_a* de la trenul aflat pe peronul *platform_a*, și le adaugă la trenul aflat pe peronul *platform_b* începând cu poziția *pos_b*. Primul vagon se află pe poziția 1. Pozițiile corecte pe care se poate insera un vagon sunt de la 1 la numărul de vagoane+1 pentru ultima poziție. (10p)

Cerința 2 (30p). Implementați următoarele funcții de căutare a trenurilor:

a). **`find_express_train(station)`** - găsește trenul cel mai rapid. Un tren se mișcă cu atât mai repede cu cât diferența între puterea locomotivei și greutatea vagoanelor este mai mare. Funcția întoarce peronul pe care se află express-ul. Se garantează unicitatea soluției. (5p)

b). **`find_overload_train(station)`** - găsește trenul care nu poate pleca din stație. Un tren nu poate pleca din stație atunci când încărcătura vagoanelor este mai mare decât puterea de tracțiune a locomotivei. Funcția întoarce peronul pe care se află trenul. Se garantează unicitatea soluției problemei. Dacă nu există un astfel de tren funcția întoarce -1. (5p)

c). **`find_optimal_train(station)`** - găsește trenul cu încărcătura optimă. Încărcătura optimă înseamnă că diferența între puterea de tracțiune a locomotivei și suma greutăților vagoanelor unui tren să fie minimă. Funcția întoarce peronul pe care se află trenul cu încărcătura optimă. Se garantează unicitatea soluției problemei. (5p)

d). **`find_heaviest_sequence_train(station, cars_no, start_car)`** - găsește trenul care are cea mai mare greutate totală pentru o secvența continuă de vagoane de lungime *cars_no*. Funcția întoarce peronul pe care se află trenul și pune în pointerul *start_car* primul vagon din secvența maximă. Dacă nu există o astfel de secvență, se va întoarce -1 pentru peron și NULL pentru primul vagon. Se garantează unicitatea soluției problemei. (15p)

Cerința 3 (30p). Implementați următoarele funcții de organizare a trenurilor:
a). **`order_train(station, platform)`** - ordonează vagoanele dintr-un tren în ordinea descrescătoare a greutăților. Funcția va modifica trenul existent. (15p)
b). **`fix_overload_train(station)`** - scoate un vagon din trenul care este supraîncărcat. Vagonul va fi ales astfel încât trenul să aibă după scoaterea vagonului încărcarea optimă dintre toate soluțiile posibile. Se garantează că există un singur tren supraîncărcat. (15p)

Pentru funcționarea corectă a funcțiilor va trebui să implementați funcția **`open_train_station(platforms_no)`** care creează o gară cu numărul de peroane egal cu parametrul *platforms_no*. De asemenea pentru a primi punctajul pentru eliberarea memoriei la sfârșitul execuției va trebui să implementați funcția **`close_train_station`**, care va elibera memoria asociată trenurilor și gării.

Pentru a putea verifica corectitudinea implementării funcțiilor voastre, va trebui să implementați o funcție simplă de afișare, **`show_existing_trains`**. Funcția va afișa într-un fișier primit ca parametru toate trenurile din gară în următorul format:

```
nr_peron: (putere_locomotivă)-|greutate_vagon1|-...-|greutate_vagonN|
```

Exemplu de output pentru funcția **`show_existing_trains`**:

```
0: (50)-|10|-|2|-|4|-|12|
1: (10)-|1|
2:
3: (100)
4:
5: (5)-|1|-|1|-|1|-|1|-|1|
```

Reguli și precizări:

- Punctajul temei va fi punctajul obținut pe cerințele 1-3 plus **10 puncte** obținute pentru eliberarea completă a memoriei. Punctele pentru eliberarea memoriei vor fi acordate doar dacă trece cel puțin un test de la cerințele 1-3. Pentru o temă implementată corect valgrind va genera următoarele mesaje:
  - *All heap blocks were freed -- no leaks are possible*
  - *ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)*
- Pentru a putea primi punctaj pe funcțiile de la cerințele 1-3 trebuiesc implementate obligatoriu funcțiile **`open_train_station`** și **`show_existing_trains`**.

- Testele verifică și cazuri limită. Asigurați-vă în fiecare funcție că toate operațiile pot fi aplicate. Spre exemplu, o locomotivă nu poate fi pusă pe un peron unde există deja un tren. Un vagon nu poate fi adăugat pe un peron unde nu există locomotivă. Nu se poate executa operația de mutare de vagoane dacă cel puțin unul din trenuri nu există.
- Este obligatorie crearea unui **README** care să conțină detaliile de implementare ale funcțiilor. În caz contrar pot exista depunctări de maxim 5 puncte.
- Este obligatoriu să scrieți codul frumos și lizibil (aka **coding style**). În caz contrar pot exista depunctări de maxim 5 puncte. În cadrul acestei teme lungimea maximă permisă a liniilor de cod este 120 de caractere.
- **Nu** se va modifica structura dată sau antetul funcțiilor din scheletul de cod al temei. **Nu** se vor folosi variabile globale pentru a reține rezultatul diferitelor funcții implementate.
- Rezolvarea temei va fi făcută exclusiv în fișierul ”*station.c*”.
- Orice implementări care trec fraudulos testele vor primi 0 pe temă indiferent de funcția în care se găsesc. Orice implementări copiate vor primi 0 pe temă indiferent de funcția în care se găsesc. O temă care **nu** compilează pe vmchecker **nu** va fi punctată.

Tema va fi încărcată pe vmchecker (vmchecker.cs.pub.ro) sub forma unei arhive zip. Arhiva va conține fișierul ”station.c” și fișierul ”README”.

Pentru a rula local checkerul folosiți comanda *./run.sh*. În cazul în care la rularea scriptului primiți mesajul ”Permission denied” folosiți comanda *chmod +x run.sh* pentru a-l face executabil.