Structuri de Date (CD 2020-2021)

Tema 2: Ierarhie

12.04 - 27.04 (deadline hard)

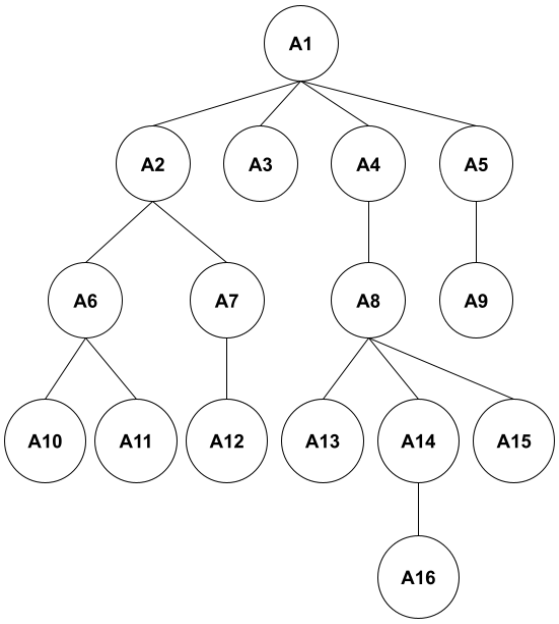
Responsabil: Alexandra Ștefania Ghiță

În această temă vom simula o ierarhie a angajaților în cadrul unei companii. Vom avea manageri, angajați și vom permite operații de transfer, reorganizare, promovări sau concedieri.

Angajații unei companii sunt dispuși într-o structură ierarhică, astfel că ei pot fi reprezentați printr-un arbore. Fiecare angajat răspunde unui manager, iar un manager poate avea unul sau mai mulți angajați. Pentru reprezentarea acestei organizații vom folosi o structură de date care reține numele angajatului și echipa pe care o conduce. Vom folosi un vector (`team`) pentru a reține nodurile următoare, întrucât un manager poate avea un număr arbitrar de angajați în echipă. Structura va avea următoarea reprezentare:

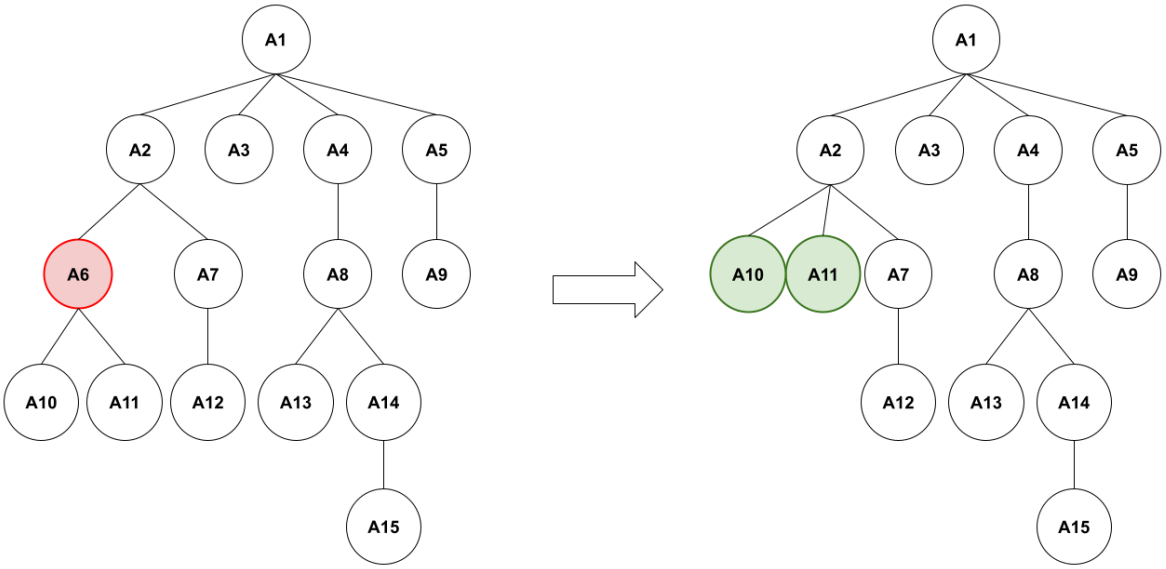
```
struct TreeNode {
	char *name;
	int direct_employees_no;
	struct TreeNode **team;
	struct TreeNode *parent;
};
```

Arborele de mai jos reprezintă o posibilă organizare a angajaților într-o firmă. Angajatul *A1* are o echipă de 4 oameni, angajatul *A8* are o echipă de 3 oameni.

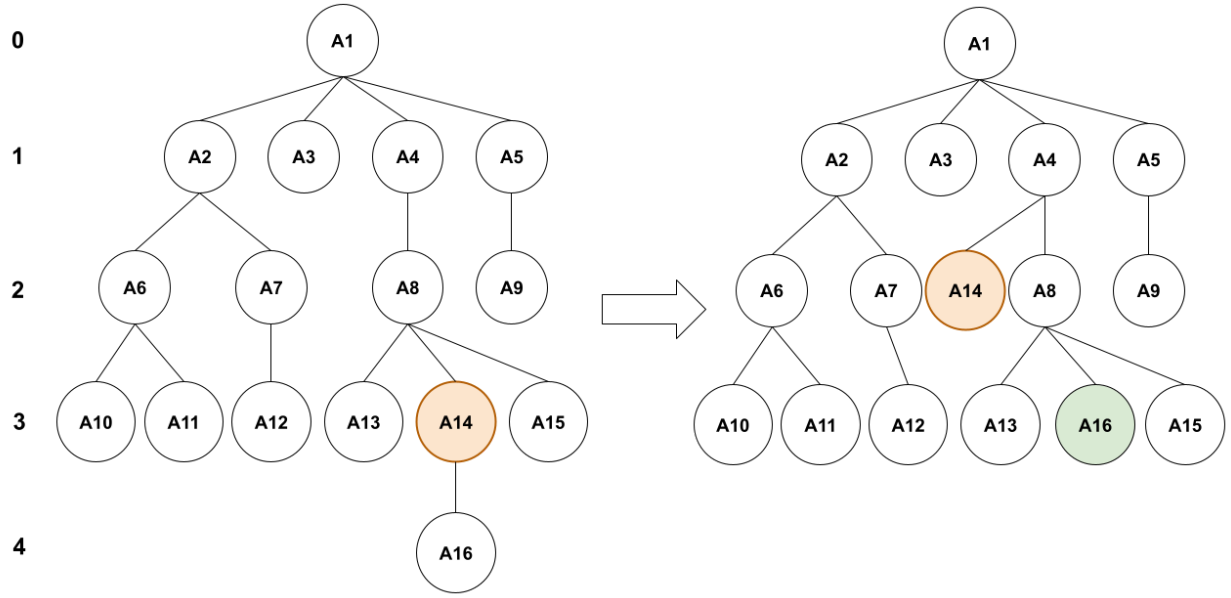


Cerința 1 (50p). În cadrul aceste metode de reprezentare a ierarhiei angajaților într-o firmă, vom permite mai multe operații care vor trebui implementate.

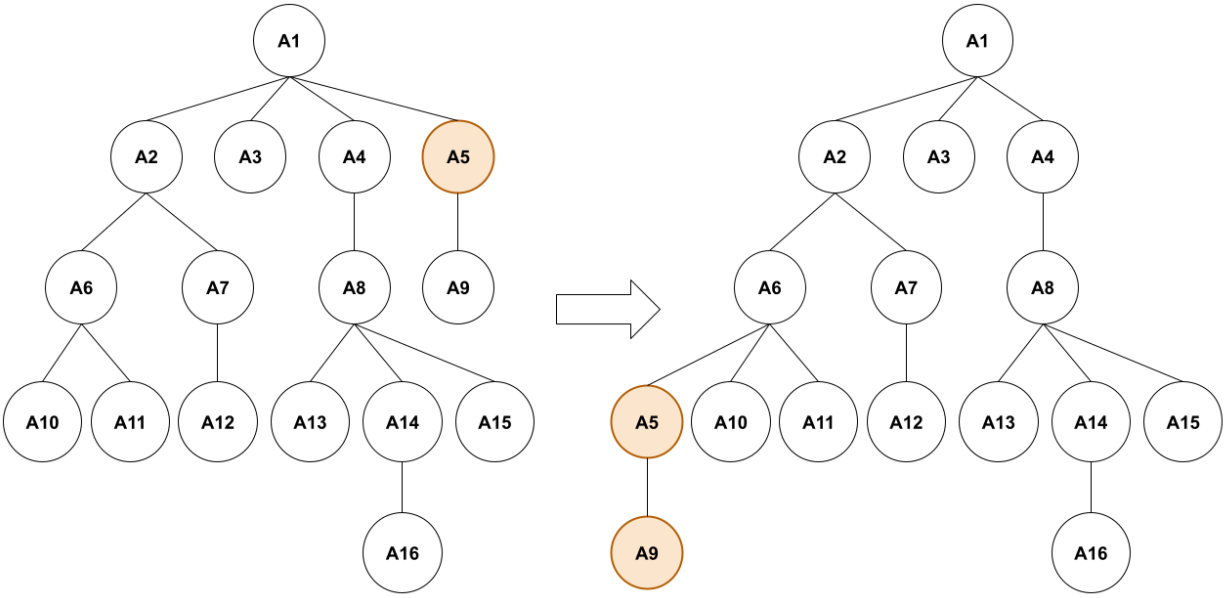
- angajarea unui nou membru al echipei (`hire`) - un angajat va fi adăugat în ierarhie. În parametrii funcției se va specifica managerul din echipa căruia va face parte. Dacă este primul nod din ierarhie, numele managerului va fi `NULL`. **(5p)**
- concedierea unui angajat (`fire`) - un angajat va fi scos din ierarhie. Toți oamenii care erau în echipa lui vor trece în echipa șefului angajatului concediat. Primul om din ierarhie nu poate fi concediat. **(10p)**



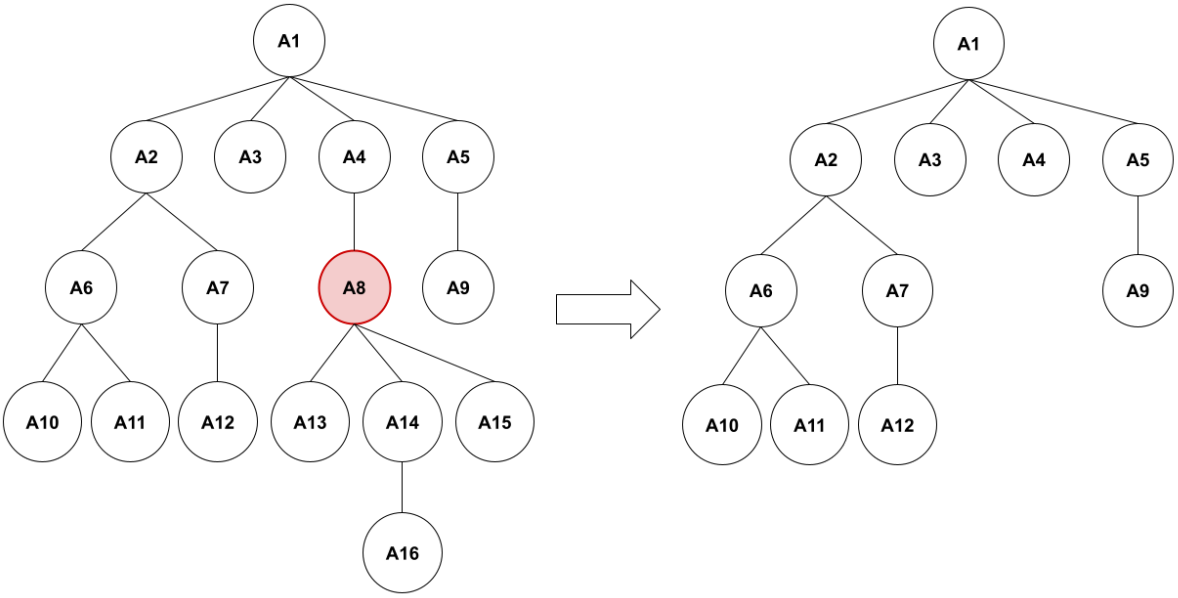
- promovarea unui angajat (`promote`) - un angajat va urca în ierarhie cu un nivel. Angajatul va merge în echipa managerului șefului curent. Echipa angajatului va merge sub conducerea șefului curent. Promovarea poate fi aplicată doar pentru angajații de la nivelul 2 în jos. **(10p)**



- mutarea unui angajat (`move_employee`) - un angajat va trece dintr-o echipă în alta. În apelul funcției va fi specificat noul manager din echipa căruia va face parte. Echipa angajatului va trece sub conducerea managerului curent. Primul om din ierarhie nu poate fi mutat. **(10p)**
- mutarea unei echipe (`move_team`) - un manager cu toți angajații lui vor trece sub conducerea unui alt manager. Primul om din ierarhie nu poate fi mutat.Se garantează că managerul nu va fi mutat sub un om aflat sub conducerea lui directă/indirectă. **(10p)**



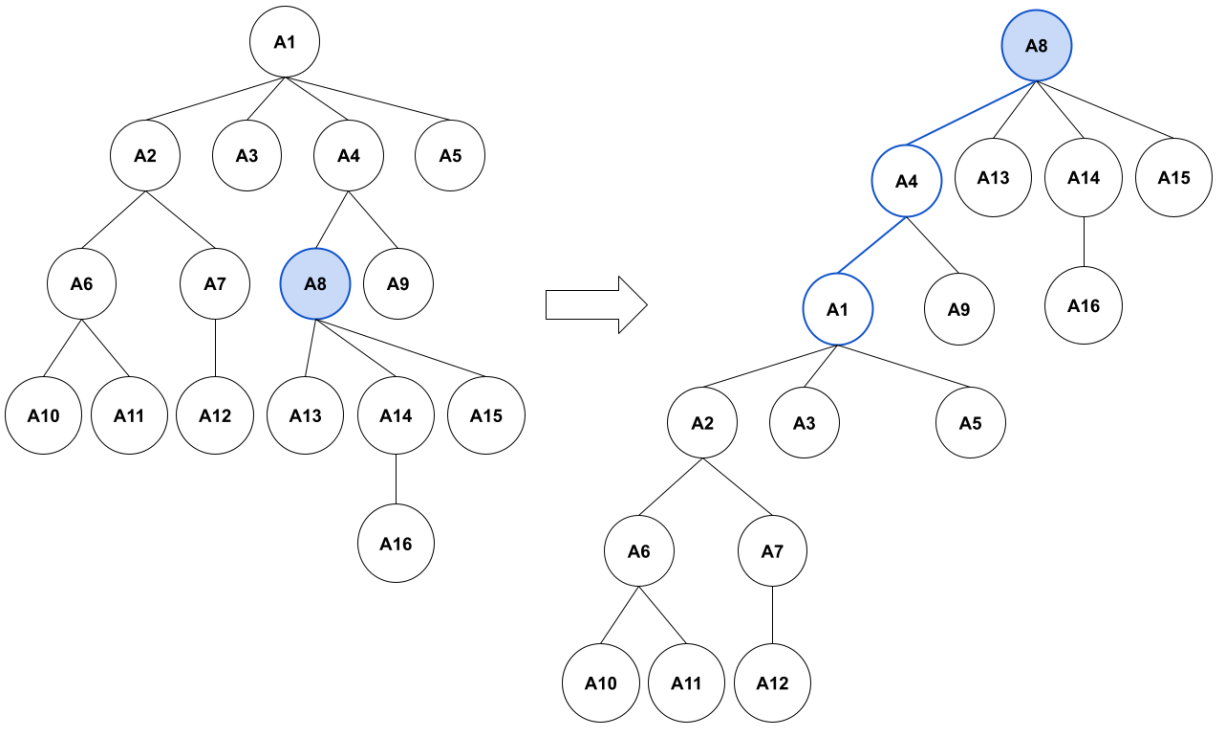
- desființarea unei echipe (`fire_team`) - un manager împreună cu toată echipa vor fi scoși din ierarhie. Primul om din ierarhie nu poate fi concediat. **(5p)**



Cerința 2 (20p). Pe această reprezentare vom implementa mai multe operații de parcurgere.

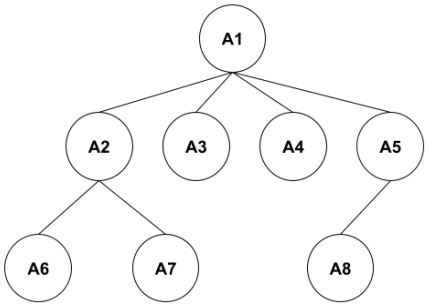
- afișarea tuturor angajaților care se află sub conducerea unui manager, direct sau indirect (`get_employees_by_manager`) - va afișa toți angajații sub conducerea unui anumit manager în ordine alfabetică. În lista afișată va fi inclus și numele managerului. **(5p)**
- afișarea tuturor angajaților de pe același nivel (`get_employees_by_level`) - va afișa toți angajații de pe un nivel dat ca parametru în ordine alfabetică. **(10p)**
- afișarea managerului cu cea mai numeroasă echipă (`get_best_manager`) - va afișa numele managerului care are cei mai mulți angajați (doar de pe nivelul imediat următor). În cazul în care sunt mai mulți manageri cu același număr de angajați se vor afișa toți în ordine alfabetică. **(5p)**

Cerința 3 (20p). În echipă se va putea face o reorganizare dacă un angajat se va demonstra că este mai bun. Implementați funcția `reorganize` prin care un angajat trece în vârful ierarhiei. Când un angajat trece în vârful ierarhiei, managerul lui va trece sub conducerea lui. Acest lucru se va aplica recursiv până la angajatul cel de mai sus. Modul în care se face reorganizarea se poate vedea în figura de mai jos. Fiecare nod aflat pe calea de la angajat la rădăcina va deveni copil pentru părintele lui.



Important. Afișarea ierarhiei este importantă pentru a evalua implementarea funcțiilor anterioare. Pentru a verifica corectitudinea arborelui vostru este **obligatorie** implementarea unei parcurgeri simple (`preorder_traversal`). Vor fi afișate numele angajaților și ale managerilor asociați (*angajat-manager*) în ordinea în care se accesează nodurile conform exemplului de mai

jos. Pentru primul om din ierarhie se va afișa doar numele angajatului. Având arborele din stânga parcurgerea generează linia din partea dreaptă.




```
A1 A2-A1 A6-A2 A7-A2 A3-A1 A4-A1 A5-A1 A8-A5
```

Reguli și precizări

- Vectorul team reține angajații direcți ai unui manager în ordine **alfabetică**.
- Angajații vor avea **nume unice** în ierarhie.
- Punctajul obținut la Cerințele 1-3 este condiționat de implementarea parcurgerii `preorder_traversal`.
- Punctajul temei va fi punctajul obținut pe checker plus **10 puncte** obținute pentru eliberarea completă memoriei. 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)*
- Este obligatorie crearea unui README care să conțină detaliile de implementare ale funcțiilor. În caz contrar pot exista depunctări de maxim 10 puncte.
- Este obligatoriu să scrieți codul frumos și lizibil (aka coding style). În caz contrar pot exista depunctări de maxim 10 puncte.
- Testele verifică și cazuri limită. Asigurați-vă în fiecare funcție că toate operațiile pot fi aplicate pe ierarhia dată.
- **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.
- 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.