Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Домашна работа

курс	*Структури от данни и програмиране*
специалност	*Компютърни науки (първи поток), група 4*
	зимен семестър 2023/24 г.

Инструкции и правила:

1. Решенията на задачите трябва да са написани на `C++` и предадени в подходящо конфигуриран `CMake` проект, който сам извлича от Интернет библиотека за `unit` тестване (например `Catch2`)
2. Решението трябва да е покрито с подходящи `unit` тестове
3. Разрешено е използването на библиотеките `<iostream>`, `<cassert>`, `<fstream>`, `<functional>`, `<stdexcept>`, `<string>`, `<vector>`, `<list>`, `<forward_list>`, `<stack>` и `<queue>`. За всяка друга библиотека, която използвате, трябва много добре да се аргументирате на защитата защо ви трябва
4. Входните данни трябва да бъдат валидирани
5. В решенията си трябва да спазвате добрите практики за именуване и чист код
6. Решения, които грубо нарушават добрите практики на ООП, ще бъдат оценявани с `0` точки
7. В заданието трябва да предадете само изходния код на решението (`.h/hpp` и `.cpp` файлове), както и проектните файлове (`CMakeLists.txt`)

Условие

Асоциацията на професионалните тенисисти (ATP) е създадена през 1972 година с цел да защитава интересите на мъжете – професионални състезатели по тенис. През 1990 година асоциацията започва да организира международна верига от тенис турнири за мъже-професионалисти, наречена ATP Tour. Разбира се, тази асоциация има много сложна йерархия от служители - шефове, мениджъри, по-големи мениджъри, по-малки

мениджъри, някои по-важни тенисисти, някои не толкова важни тенисисти и т.н. Баш CEO-то на ATP обаче е само един, но за улеснение ще му викаме просто “CEO_to”. За да успява да следи клоновата си мрежа и всички нейни служители, CEO-то помолил своя екип разработчици да му напише програма, която да може да прочита от текст (string) описание на професионални взаимоотношения и след това да построява дърво, което да представя тези връзки (шеф-подчинен). Едно такова дърво описва взаимоотношенията в рамките на един клон на компанията (отдел за финанси, отдел за организация на турници, т.н.). Понеже преди да се захване с този бизнес, CEO-то бил учил малко във ФМИ, той знаел, че когато всеки служител (освен него самия, разбира се) има точно един шеф, то построяване на такова дърво било не само възможно, но и лесно.

Построяване на йерархията

Програмата ви ще трябва да може да зарежда входа си от текстов файл. Всеки един такъв файл съдържа нула, един или повече редове, като всеки ред съдържа описанието на едно взаимоотношение ръководител-подчинен. Например редът “Federer - Grisho” означава, че Федерер е пряк ръководител на Гришо.

Имената на хората не съдържат празни символи или тире и всяка такава двойка е на отделен ред. Редът, в който те са подредени при въвеждането не е строго определен, но трябва всеки с роля ръководител първо да се е появил с роля подчинен (освен главният изпълнителен директор, който на никого не е подчинен). Например не може във входа да се срещне ред “Federer - Grisho”, ако преди това не е имало друг запис, в който да се укаже на кого е подчинен Федерер.

Всеки човек се предполага да се среща точно един път в ролята на подчинен в записите. За улесняване на задачата предполагаме, че не може да има двама човека с еднакви имена, като отчитаме разликите между малки и главни букви. Тъй като предполагаме, че в корена на всяко едно такова дърво стои CEO-то, за него не добавяме запис - той може да бъде използван директно, за да се укаже кои са преките му подчинени, например “CEO_to - 6ef_na_parite”.

Няма ограничение в дълбочината на дървото, което ще се построи.

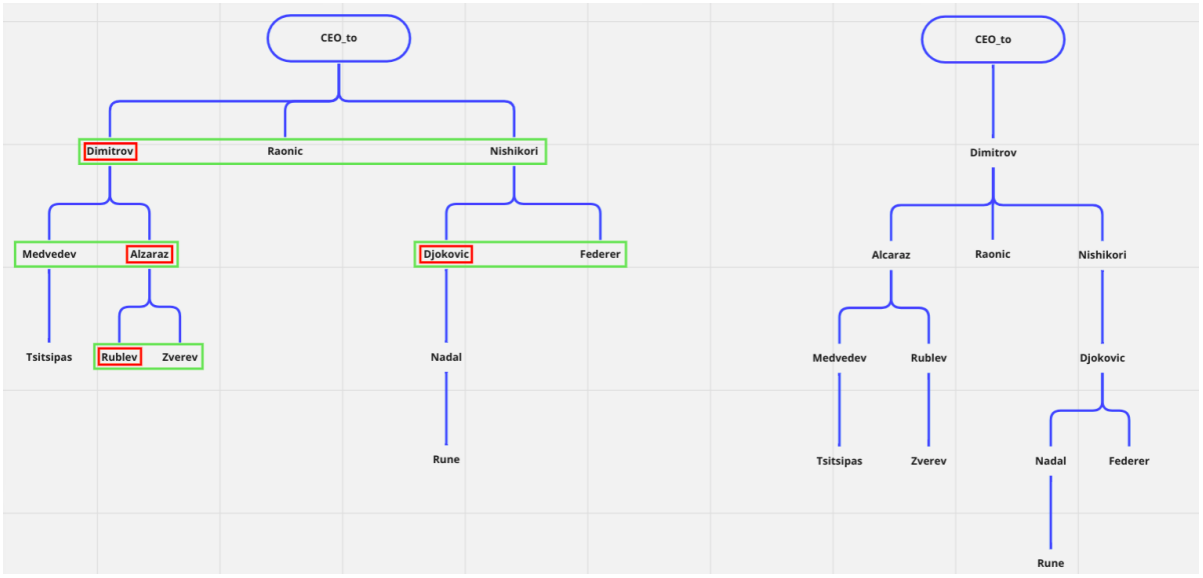
Операции

Освен да зареждате такива данни и да построявате съответните на йерархиите им дървета трябва да можете да извършвате различни операции с тях:

- На първо място трябва да проверите дали подадените ви данни са коректни и ако не са, да сигнализирате за това;
- Да можете да проверите дали даден служител е част от дадена йерархия;
- За даден служител трябва да можете да кажете броя на преките му подчинени;
- За даден служител трябва да можете да кажете името на прекия му ръководител;

- За дадена йерархия трябва да можете да кажете броя на всички служители в нея;
- Трябва да намирате броя на всички ръководители, които са претоварени - имат повече от N подчинени (преки или не). N е параметър на операцията;
- Трябва да може да обединявате две дървета (представящи йерархии от два отдела). Това да става по следната схема:
    - Ако служител се среща и в двете йерархии, то в обединението той също се среща като там преките му подчинени са всички негови преки подчинени от двете йерархи. Предполага се CEO-то да е на върха и на двете места. Това правило се прилага рекурсивно надолу (за всички служители от върха надолу). Ако в двата отдела този служител има различни ръководители, то в обединението трябва да остане само една инстанция, закачена към по-високо стоящия в йерархията ръководител. Ако двата са на едно ниво (например преки подчинени на CEO-то), то изберете този с лексикографски по-малкото име. Не се предполага за двама служители в едната йерархия единият да е подчинен на другия, а в другата - обратно;
    - Ако даден служител се среща само в едната йерархия, той присъства в обединението, точно под инстанцията на ръководителя си (като негов пряк подчинен), заедно с всички свои подчинени от съответното дърво;
    - Сливането не е възможно ако има ситуация, в която служител X е подчинен на Y в едната йерархия (пряко или не), а в другата е обратно. В такъв случай не трябва да създавате слятата йерархия.
- По подадена йерархия и име на служител трябва да уволните (премахнете) служителя, като всички негови подчинени стават съответно подчинени на ръководителя му. Разбира се, няма как да се уволни CEO-то;
- По подадена йерархия, име на служител и име на ръководител трябва да назначите служителя като подчинен на този ръководител. Проверете дали служителят вече не работи с друг ръководител. Тогава преместете служителя на новата му позиция (преназначаване). Ако в този случай служителят е имал подчинени, то те остават под него в йерархията;
- Трябва да можете да записвате дърво в символен низ (string). Форматът е описан по-подробно в следващия раздел на документа;
- За една фирма трябва да можете да кажете от колко човека се състои най-дългата верига от отношения ръководител-подчинен;
- Трябва да може да смятате заплатата на даден служител, като тя се определя по следната формула: 500 * <брой преки подчинени> + 50 * <брой непреки подчинени>;

- Трябва да може да инкорпорирате една йерархия. Това става като за всеки екип (множество служители с общ пряк ръководител), който има поне двама служители, се избере служителят с най-висока заплата и се направи шеф на този екип. Ако този служител има собствени подчинени, те остават в неговия екип. Ако има повече от един такъв служител, да се избере този с лексикографски най-малкото име. Инкорпорирането започва от най-ниските нива в йерархията. На фигурата по-долу е показана йерархия преди инкорпориране и съответно след. В първоначалната йерархия са заградено със зелено екипите и с червено служителят, който ще бъде повишен.



- Трябва да може да модернизирате една фирма. Това става като за всеки ръководител на нечетно ниво спрямо CEO-то, неговият екип се слее с екипа на по-висшия ръководител. Ръководителят се премахва. Модернизацията започва от най-ниските нива на йерархията.

Представяне като символен низ

Всяко дърво от разглеждания тип може да се представи като редица от двойки от вид *ръководител-подчинен*. В редицата те трябва да са подредени по следния начин:

1. Старшинство (от най-висшия към по-нисшите ръководители).
2. Записите, в които има двама ръководители от едно ниво на йерархията, се подреждат лексикографски – най-напред по името на ръководителя, а когато то съвпада за два различни записа, по името на служителя.

Всяка двойка ръководител-подчинен се представя като низ като конкатенираме техните имена със символа тире и завършва със символа за нов ред. Дървото

представяме като конкатенация на всички такива двойки. Например за лявото дърво от фигурата по-горе ще имаме следното представяне:

CEO_to-Dimitrov
CEO_to-Raonic
CEO_to-Nishikori
Dimitrov-Alcaraz
Dimitrov-Medvedev
Nishikori-Djokovic
Nishikori-Federer
Alcaraz-Rublev
Alcaraz-Zverev
Medvedev-Tsitsipas
Djokovic-Nadal
Nadal-Rune

Команди

Програмата, която реализирате, трябва да има интерактивен конзолен потребителски интерфейс, в който да се поддържат всички гореизброени функционалности. Ако нещо не е наред (грешен формат на данните, не е намерен посочен файл и др.) програмата ви трябва да продължи да работи коректно и трябва да изведете на екрана подходящо съобщение за потребителя.

Командите, които трябва да обработвате (като минимум) са:

- `help` – извежда списък на поддържаните команди с кратка помощна информация за тях;
- `load <име_на_обект> <име_на_файл>` – зарежда данни за йерархия от файл с подаденото име и създава дърво, асоциирано с име_на_обект. Това име трябва да се състои само от малки и главни латински букви, цифри и символ за подчертаване. След него всичко до края на реда е името на файла от който трябва да прочетете данните в описания по-горе формат. Ако името на файла липсва се предполага да прочетете данните от стандартния вход, до срещане на край на файл (Ctrl+z/Ctrl+d). Ако не можете да се справите с това, четете до празен ред;
- `save <име_на_обект> <име_на_файл>` – записва информацията за йерархията на посочения обект във файл с посоченото име. Ако името на файла е празно, информацията да се изведе на стандартния изход;
- `find <име_на_обект> <име_на_служител>` – проверява дали в посочения обект съществува служител с посоченото име;
- `num_subordinates <име_на_обект> <име_на_служител>` – извежда броя преки подчинени на дадения служител в посочения обект;

- `manager <име_на_обект> <име_на_служител>` – извежда името на ръководителя на дадения служител в посочения обект;
- `num_employees <име_на_обект>` – извежда броя служители в посочения обект;
- `overloaded <име_на_обект>` – извежда броя служители в посочения обект, за които броят подчинени (преки или не) е по-голям от 20;
- `join <име_на_обект_1> <име_на_обект_2> <име_на_обект_резултат>` – обединява двата подадени обекта в нов обект с име <име_на_обект_резултат>;
- `fire <име_на_обект> <име_на_служител>` – премахва служителя от съответния обект;
- `hire <име_на_обект> <име_на_служител> <име_на_ръководител>` – назначава служителя в съответния обект като подчинен на подадения ръководител;
- `salary <име_на_обект> <име_на_служител>` – извежда заплатата на служителя;
- `incorporate <име_на_обект>` – инкорпорира фирмата; операцията се прилага върху обекта <име_на_обект>;
- `modernize <име_на_обект>` – модернизира фирмата; операцията се прилага върху обекта <име_на_обект>;
- `exit` - прекратява изпълнението на програмата. За всички нови или променени след зареждането обекти попитайте потребителя дали иска да ги запази във файл.

Примерно изпълнение на програмата

```
> load ATP
CEO_to-Dimitrov
CEO_to-Raonic
CEO_to-Nishikori
Dimitrov-Alcaraz
Dimitrov-Medvedev
Nishikori-Djokovic
Nishikori-Federer
Alcaraz-Rublev
Alcaraz-Zverev
Medvedev-Tsitsipas
Djokovic-Nadal
Nadal-Rune
^Z
ATP loaded successfully!
```

```
> find ATP Tsitsipas
Tsitsipas is employed in ATP.

> num_subordinates ATP Tsitsipas
Tsitsipas has no subordinates.

> num_subordinates ATP Nishikori
Nishikori has two subordinates.

> manager ATP Djokovic
The manager of Djokovic is Nishikori.

> manager ATP Ruud
There is no Ruud in ATP.

> num_employees ATP
There are 13 employees in ATP.

> num_employees ATP2
ATP2 is an unknown office!

> overloaded ATP
No overloaded employees in ATP.

> load ATP_new
CEO_to - Sinner
Sinner - Raonic
Sinner - Ruud
^Z
ATP_new loaded successfully!

> join ATP ATP_new ATP_big
ATP_big created.

> save ATP_big ATP_big.txt
ATP_big saved.

> manager ATP_big Raonic
The manager of Misho is CEO_to.

>num_subordinates ATP_big Sinner
Sinner has 1 subordinats.

> fire ATP_big Sinner
Sinner was fired.

> num_subordinates ATP_big CEO_to
CEO_to has 4 subordinates.

> save ATP_big ATP_big.txt
ATP_big saved.
```

> hire ATP Sinner Misho
Sinner was hired.

> salary ATP Dimitrov
The salary is 1150 euro.

> hire ATP_new Hurkacz CEO_to
Hurkacz *was hired.*

> hire ATP_new *Sinner* Hurkacz
Sinner was hired.

> incorporate ATP_new
ATP_new incorporated.

> save ATP_new
CEO_to-Hurkacz
Hurkacz-Sinner
Sinner-Raonic
Raonic-Ruud

> hire ATP_new Fritz Raonic
Fritz *was hired.*

> hire ATP_new Monfils Fritz
Monfils *was hired.*

> modernize ATP_new
ATP_new *modernized.*

> save ATP_new
CEO_to-Sinner
Sinner-Fritz
Sinner-Ruud
Fritz-Monfils

> save ATP_new ATP_new.txt
ATP_new *saved.*

> exit
ATP is modified, but not saved.
Enter file name to save it:
> ATP.txt
ATP saved.
Goodbye!