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

Второ домашно

курсове *Структури от данни;*
Структури от данни и програмиране

специалности *Информатика, Информационни системи и*
Компютърни науки (1-ви поток)

зимен семестър 2021/22 г.

Редакции

18.12 - Поправка в примерите за инкорпориране и модернизиране.

Условие на задачата

След невероятния успех на кампанията по продажба на банани и швепс в магазините МразМаг фирмата се разраснала неимоверно. Толкова много, че се наложило да въведат много сложна йерархия на служители, по-малки мениджъри, по-големи мениджъри и шефове. Разбира се главният изпълнителен директор останал един – Петър Петров (по прякор Успешния).

За да успява да следи клоновата си мрежа и всички нейни служители, Успешният 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, магазин и т.н.). Понеже преди да се захване с търговия на дребно, Успешния 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.

Вход

Програмата ви ще трябва да може да зарежда входа си от един или повече текстови файлове. Всеки един такъв файл съдържа нула, един или повече редове, като всеки ред

съдържа описанието на едно взаимоотношение ръководител-подчинен. Например редът “Ivan-Georgi” означава, че Иван е пряк ръководител на Георги.

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 “Ivan-Georgi”, ако преди това не е имало друг запис, в който да се укаже на кого е подчинен Иван.

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 Успешния, за него не добавяме запис - той може да бъде използван директно, за да се укаже кои са преките му подчинени, например “Uspeshnia - Ivan”.

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 Успешният 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 Успешния), 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 Успешния;

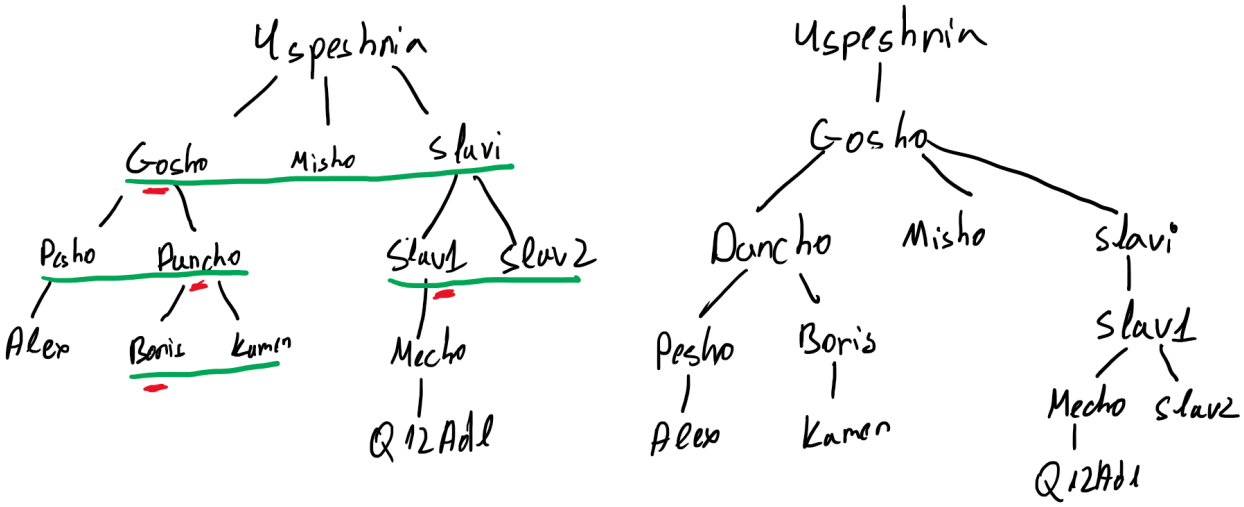
- По подадена йерархия, име на служител и име на ръководител трябва да назначите служителя като подчинен на този ръководител. Проверете дали служителят вече не работи с друг ръководител. Тогава преместете служителя на новата му позиция (преназначаване). Ако в този случай служителят е имал подчинени, 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 не преки подчинени>;

- Трябва да може да инкорпорирате една йерархия. Това става като за всеки екип (множество служители с общ пряк ръководител), който има поне двама служителя, 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 подчертани 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 Успешния, неговият екип се слее с екипа на по-висшия ръководител. Ръководителят се премахва. Модернизацията започва от най-ниските нива на йерархията.

Представяне на дърво 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:

```
"Uspeshnia-Gosho\nUspeshnia-Misho\nUspeshnia-Slavi\nGosho-Dancho\nGosho-Pesho
nSlavi-Slav1\nSlavi-Slav2\nDancho-Boris\nDancho-Kamen\nPesho-Alex\nSlav1-Mech
o\nMecho-Q12Adl\n"
```

Което, за по-лесно, даваме по-долу и така както би изглеждал след извеждане на екрана:

```
Uspeshnia-Gosho
Uspeshnia-Misho
```

```
Uspeshnia-Slavi
Gosho-Dancho
Gosho-Pesho
Slavi-Slav1
Slavi-Slav2
Dancho-Boris
Dancho-Kamen
Pesho-Alex
Slav1-Mecho
Mecho-Q12Adl
```

Операции

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);
- `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 Lozenec
Uspeshnia - Gosho
Uspeshnia - Misho
Gosho - Pesho
Gosho - Dancho
Pesho - Alex
Dancho-Boris
Dancho-Kamen
Uspeshnia - Slavi
Slavi - Slav1
Slavi - Slav2
Slav1-Mecho
Mecho-Q12Adl
^Z
Lozenec loaded successfully!

> find Lozenec Alex
Alex is employed in Lozenec.

> num_subordinates Lozenec Alex
Alex has no subordinates.

> num_subordinates Lozenec Slavi
Slavi has two subordinates.

> manager Lozenec Slav1
The manager of Slav1 is Slavi.

> manager Lozenec Slav3
There is no Slav3 in Lozenec.

> num_employees Lozenec
There are 13 employees in Lozenec.

> num_employees Lozenec2
Lozenec2 is an unknown office!

> overloaded Lozenec
No overloaded employees in Lozenec.

> load Lozenec_new
Uspeshnia - MishoPetrov
MishoPetrov - Misho
MishoPetrov - Slav
^Z
Lozenec_new loaded successfully!
```

```
> join Lozenec Lozenec_new LozBig
LozBig created.

> save LozBig LozBig.data
LozBig saved.

> manager LozBig Misho
The manager of Misho is Uspeshnia.

>num_subordinates LozBig MishoPetrov
MishoPetrov has one subordinates.

> fire LozBig MishoPetrov
MishoPetrov was fired.

> num_subordinates LozBig Uspeshnia
Uspeshnia has four subordinates.

> save LozBig LozBig.data
LozBig saved.

> hire Lozenec MishoPetrov Misho
MishoPetrov was hired.

> salary Lozenec Gosho
The salary is 1150 BGN.

> hire Lozenec_new Mitko Uspeshnia
Mitko was hired.

> hire Lozenec_new MishoPetrov Mitko
MishoPetrov was hired.

> incorporate Lozenec_new
Lozenec_new incorporated.

> save Lozenec_new
Uspeshnia-Mitko
Mitko-MishoPetrov
MishoPetrov-Misho
Misho-Slav

> hire Lozenec_new Ivan Misho
Ivan was hired.

> hire Lozenec_new Stojan Ivan
Stojan was hired.

> modernize Lozenec_new
Lozenec_new modernized.

> save Lozenec_new
Uspeshnia-MishoPetrov
MishoPetrov-Ivan
MishoPetrov-Slav
Ivan-Stojan

> save Lozenec_new loz.data
Lozenec_new saved.

> exit
Lozenec is modified, but not saved.
Enter file name to save it:
> lozenec.data
Lozenec saved.
Goodbye!
```

Тестове

Тестове за задачата ще намерите в хранилището на следния адрес https://github.com/semerdzhiev/sdp-2021-22/tree/main/tests/2. Вашето решение трябва като минимум да минава тези тестове. Ако решението ви минава всички тестове това не означава автоматично получаване на пълен брой точки. Тестовете проверяват няколко конкретни случая и не са изчерпателни. Препоръчваме да добавите свои такива, които обхващат повече случаи.

За да може тестовете да проверяват коректността на вашето решение трябва да спазите няколко условия при реализация на кода си.

В interface.h ще намерите класа **`Hierarchy`**, който описва интерфейса, който вашето решение трябва да спазва за да може да бъде тествано от автоматичните тестове. Трябва да имплементирате всички изброени методи в реализация на този клас.

В tests.cpp ще намерите всички тестове, които вашето решение трябва да минава успешно. Тестовете трябва да минават без промяна на кода в този файл.
Ако смятате че някой тест противоречи на условието, пишете в дискорд канала и **тагнете някой от екипа**.

ВАЖНО: Не забравяйте, че решението на задачата трябва да може да чете данни от клавиатурата и да пише резултата обратно в конзолата. Същото това решение трябва и да може да бъде тествано с автоматичните тестове. Това означава че трябва да има 2 под проекта (project във Visual Studio) - единият да пуска тестовете, а другият да имплементира вход/изход от потребител.

При реализацията е разрешено да използвате само класовете `string`, `vector`, `list`, `forward_list`, `stack` и `queue` от стандартната библиотека. Всички останали структури от данни или алгоритми, които са ви необходими при решаването на задачата трябва да реализирате сами.