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

Домашна работа 1

курс Структури от данни и програмиране
за специалност Информатика
зимен семестър 2018/19 г.

По случай началото на учебната година Интегралчо решил да покани приятелите си за едно последно събиране преди „играта да загрубее“. Интегралчо бил социално момче и съответно имал приятели от различни университети. Той знае, че най-много трябва да внимава за студентите от Техническия университет (ТУ), Университета за национално и световно стопанство (УНСС) и Факултета по математика и информатика (ФМИ) към Софийския университет, когато са събрани на едно място. Техните отношения са сложни и са нужни грижи, за да не се стига до „инциденти“. Това било особено важно, понеже Интегралчо планирал гвоздеят на програмата да бъде разпространилият се в Куба (въпреки, че нашият герой не знае това) танц “конга”. Той прилича на опашка от хора, хванали се един за друг, всеки за човека пред себе си. Поучил се от предшествениците си, Интегралчо знае, че не може да остави нещата на случайността и затова ви моли да му помогнете, като напишете програма, която да следи позициите на хората, които участват в танците и цели да предотврати „нежелани“ комбинации.

Отношенията между гореспоменатите студенти най-често следвали следния принцип: студентите от ФМИ не харесвали отпуснатите и според тях „разхайтени“ колеги от УНСС и изпитвали симпатии към тези от Техническия университет, като братя с подобна съдба. Тези от ТУ пък намирали колегите си от ФМИ за претенциозни елитисти, но се радвали на „свободния дух“ на тези от УНСС. Те на свой ред се подигравали на ТУ и измисляли груби вицове за момичетата им и одобрявали колегите от ФМИ, поради тесния кръг от хора от там, с които са се сприятелили по дискотеките. Студентите от всеки университет се разбирали помежду си. Накратко:

- fmi толерират: fmi и tu
- tu толерират: tu и unss (unwe, но unss за целите на задачата)
- unss толерират: unss и fmi

Вие трябва да напишете програма, която поддържа множество конга опашки, в които пред никого не стои човек от университет, който той/тя не харесва. По-долу са дадени примери за различни опашки от хора от различни университети, заедно с коментар дали те са валидни или не:

№	Ред на хората	Коментар
1.	`fmi <– unss <- tu <- tu <- fmi`	Валидна конга опашка
2.	`tu <- fmi <- fmi <- unss <- tu`	Валидна конга опашка
3.	`tu <- fmi <- tu`	Не е валидна, понеже човек от ФМИ е пред човек от ТУ

Всеки човек в опашката се характеризира със своето име и университет/факултет.

Програмата ви трябва да започне с една празна опашка и да поддържа дадените по-долу команди. Когато програмата се стартира, тя позволява на потребителя да въвежда командите в произволен ред. След изпълнението на всяка от тях трябва да се извежда кратко съобщение, което показва как е приключила. Излизането от програмата става с командата `quit`.

По-долу е дадено описание на командите. Считаме, че `NAME` и `UNI` са символни низове, а `LINEINDEX[1|2]` е число.

`append NAME UNI LINEINDEX`
Добавя човек с име NAME и университет/факултет UNI към опашка с индекс LINEINDEX (всяка опашка се характеризира със своя индекс), но само ако пред новия човек има друг от съвместим с неговия университет. В противен случай изведете на стандартния изход съобщение за грешка “Incompatible people”.

`removeLast LINEINDEX`
Премахва последния участник от опашката с индекс LINEINDEX. Ако опашката се изпразни, тя се изтрива. В резултат от операцията, останалите опашки може да променят индексите си.

`removeFirst LINEINDEX`
Премахва първия участник от опашката с индекс LINEINDEX. Ако опашката се изпразни, тя се изтрива. В резултат от операцията, останалите опашки може да променят индексите си.

`remove NAME LINEINDEX`
Премахва първия срещнат (броено отпред-назад, започвайки от „водача“) човек с име NAME от опашката с индекс LINEINDEX. Ако човекът не е краен („водач“ или най-отзад), старата опашка трябва да остане същата до преди неговата позиция, а хората след неговата позиция да образуват нова опашка. Ако някоя опашка остане празна след премахване на човек от нея, тя трябва да бъде премахната от колекцията. В резултат от операцията, останалите опашки може да променят индексите си.

`merge LINEINDEX1 LINEINDEX2`
Обединява опашките с индекси LINEINDEX1 и LINEINDEX2, но само ако последният човек от първата и първият от втората са със съвместими университети (правилото отново е, че всеки човек гледа този пред себе си; първият човек от втората опашка не трябва да е несъвместим с последният от първата). В противен случай операцията не прави нищо и извежда съобщение "Incompatible people".

`print`
Извежда опашките в следния формат:

`Line0: (name-00, uni-00) - (name-01, uni-01) ⋯ (name-0`m_0`, uni-0`m_0`)`
`⋯.`
`Linen: (name-n0, uni-n0) - (name-n1, uni-01) ⋯ (name-nm`$_n$`, uni-nm`$_n$`)`

Където name-xy е името на човека на x-тата опашка, на място y. Аналогично за uni-xy. n е броят на опашките. m_i е броят на елементите в опашката с индекс i. Например:

```
Line0: (dragan, tu) – (petkan, fmi) – (joro, unss) – (georgi, unss)
Line1: (aleksander, unss)
```

`quit`
Прекратява изпълнението на програмата

ВАЖНО: Изберете подходящи структури от данни, с които да представите всяка от опашките, а също и множеството от всички опашки. Изборът ви трябва да е такъв, че операциите да могат да се изпълняват с възможно най-добра сложност. За целите на задачата, считаме, че изборът ще бъде измежду базовите контейнери: динамичен масив, списък с една връзка, двойносвързан списък.