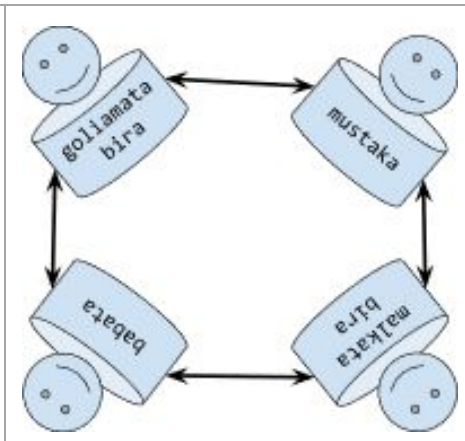
Чорбаджийско хоро

В българския фолклор се среща описанието на един интересен танц – чорбаджийско хоро. При него участниците са наредени в кръг и всеки от тях държи за коланите тези до него. В резултат на това, когато даден участник се опита да излезе от хорото, това няма как да стане, докато другите двама го държат. Вярно е и обратното – може да не е лесно да бъде "премахнат" даден участник от хорото, защото дори и съседите му да го пуснат, той може да се държи здраво за тях.

В тази задача ще трябва да моделирате горе-описаната ситуация с подходяща структура от данни. Програмата ще получава като параметър от командния ред път до текстов файл. Този файл ще съдържа N на брой символни низа ($N \geq 3$), всеки от които на отделен ред. Считаме, че тези низове представят участниците в хорото – например техни имена или прякори и т.н. Низовете са подредени във файла точно в реда, в който се държат участниците в хорото, като за да се получи кръг, считаме, че последният държи първия. За улеснение считаме, че всеки етикет е с дължина не повече от 30 символа. В хорото не може да има двама участника с един и същ етикет. По-долу са дадени примерен входен файл и хорото, което той описва (забележете, че в имената може да има интервали):

```
goliamata bira
mustaka
malkata bira
babata
```



След като зареди данните за хорото, програмата трябва да въвежда и изпълнява команди. Те са описани по-долу. Всяка от командите, освен PRINT и EXIT трябва да работи със сложност O(1). PRINT трябва да бъде със сложност O(N). За EXIT няма ограничение. Решението ви трябва да използва контейнерите от STL. Тъй като задачата цели да упражни използването на STL и стандартната библиотека на C++, постарайте се да не пишете собствен код, ако има достатъчно добра функция или клас от стандартните. В задачата не бива да се използват външни библиотеки (например Boost).

`RELEASE <who> [left|right|both]`

Прави така, че участникът с етикет <who> да пусне левия (left) или десния (right) си съсед или и двамата (both).

`GRAB <who> [left|right|both]`

Прави така, че участникът с етикет <who> да хване левия (left) или десния (right) си съсед или и двамата (both).

`INFO <who>`

Извежда на екрана информация за участника <who>, в следния формат:

```
<left> <relation-left> <who> <relation-right> <right>
```

където:

- `<who>` е етикетът на участника;
- `<left>` и `<right>` са етикетите на левия и десния съсед;
- `<relation-left>` и `<relation-right>` са дъги, чиито краища показват кой-кого държи.

Например:

`a <--> b ---> c`	a и b се държат един за друг; b държи c, но c е пуснал b
`a ---> b <--- c`	a и c държат b, но b ги е пуснал
`a ---- b ---- c`	никой не държи никого

`ADD <who> <label-left> <label-right>`

Добавя нов участник с етикет <who> в хорото. Той се добавя между участниците с етикети <label-left> и <label-right>, но само ако те са съседи. Ако не са, операцията извежда съобщение за грешка и не прави нищо. При добавянето, новият участник се хваща за съседите си, а те също го хващат.

`REMOVE <who>`

Премахва участника с етикет <who> от хорото, но само ако той е пуснал и двамата си съседи и те също са го пуснали. Ако премахването е успешно, да се изведе текст *"Free at*

last!", а в противен случай – *"This won't be so easy!"*. Ако хорото остане само с двама участника, то се "разтурва" и танцът приключва. В този случай програмата трябва да изведе съобщение *"...and the music stops!"* и да прекрати своето изпълнение.

`SWAP <who1> <who2>`

Разменя местата на участниците с етикети <who1> и <who2> в хорото, но само ако:

1. те са съседи;
2. не държат за коланите никой друг участник и никой друг участник не държи тях (двамата обаче може и да се държат помежду си).

В противен случай, командата не прави нищо. Тя трябва да бъде комутативна и потребителят може да подава етикетите в произволен ред (тоест SWAP A B трябва да бъде еквивалентно на SWAP B A).

`PRINT`

Извежда на екрана всички участници в хорото, в реда, в който се държат. Всеки участник се извежда на отделен ред. Форматът е същият като на входния файл.

`EXIT`

Излиза от програмата. При излизането всички използвани ресурси трябва да се освобождават коректно. Въпреки, че няма ограничение за сложността на EXIT, тя не трябва да бъде излишно утежнена.

Фаза 1

На тази фаза от проекта трябва да разработим дизайна на решението. За целта ще бъде организирани в екипи от двама или трима души. Всеки екип трябва да създаде модел на решение. Той трябва да бъде описан с подходящи UML диаграми. Като минимум трябва да имате една или повече клас диаграми, които показват организацията на решението. Вероятно ще се наложи да изготвите и диаграми, които описват поведението (например Sequence и/или Activity и т.н.). Организирайте решението си в MVC или MVP стил.

Hint: използвайте design patterns като например Observer, Builder и/или Factory.