

Zadanie D*

Zabawa karnawałowa

Wieść niesie, że jeszcze przed rozpoczęciem nowego semestru zajęć, podczas jednej z hucznych zabaw karnawałowych uczestnicy bawili się formując się w pociągi, z których każdy ma swoją nazwę. Imprezę prowadził znany konferansjer Maciej, który komentował zabawę sali, a także wprowadzał dodatkowy element do zabawy w postaci poleceń, które karnawałowicze mieli wykonywać.

Twoim zadaniem jest przeprowadzić symulację zabawy. Źródłem informacji dla Ciebie jest chronologiczna lista wypowiedzi Pana Macieja dotycząca imprezy:

- **NEW Train1 Person** — oznacza iż osoba **Person** tworzy nowy pociąg o nazwie **Train1**.
- **BACK Train2 Person** — oznacza iż osoba **Person** dołącza na koniec pociągu o nazwie **Train2**.
- **FRONT Train2 Person** — oznacza iż osoba **Person** dołącza na początek pociągu o nazwie **Train2**.
- **PRINT Train2** — oznacza, że musisz wypisać na wyjściu opis pociągu o nazwie **Train2**.
- **REVERSE Train2** — oznacza, iż pociąg o nazwie **Train2** zawraca.
- **UNION Train2 Train3** — oznacza, iż pociąg o nazwie **Train3** dołącza na końcu pociągu o nazwie **Train2** i przestaje istnieć.
- **DELFRONT Train1 Train2** — oznacza, iż pierwsza osoba z pociągu o nazwie **Train2** odłącza się i tworzy samodzielnie pociąg o nazwie **Train1**. Jeśli była to jedyna osoba to **Train2** przestaje istnieć.
- **DELBACK Train2 Train1** — oznacza, iż ostatnia osoba z pociągu o nazwie **Train2** odłącza się i tworzy samodzielnie pociąg o nazwie **Train1**. Jeśli była to jedyna osoba to **Train2** przestaje istnieć.

Możesz założyć, że w trakcie trwania zabawy liczba równocześnie istniejących pociągów nie przekroczy 20. Wszystkie wymienione operacje (poza **PRINT**) muszą działać w czasie $O(1)$ i używać jak najmniej pamięci. Możesz też założyć, że wszystkie polecenia są sensowne, tzn. Pan Maciej nie utworzy drugiego pociągu o tej samej nazwie, ani też nie wywoła do łączenia czy odwracania nieistniejącego pociągu.

Zadanie należy zrealizować przez zaimplementowanie dwóch szablonów `node<T>` i `doubleList<T>`. Klasa `doubleList<T>` reprezentuje listę wskaźnikową podwójnie związaną. Zawiera ona pole typu `string` pamiętające nazwę listy oraz dwa wskaźniki: na

pierwszy i ostatni element listy. Klasa `node<T>` reprezentuje jeden element listy. Zawiera pole typu `T` oraz dwa wskaźniki: na elementy sąsiadujące na liście.

Klasa `doubleList<T>` udostępnia między innymi poniższe metody:

- `void addFirst(T& name)` – dodaje nowy element zawierający `name` na początek listy.
- `void addFirst(node<T>* n)` – dodaje element `n` na początek listy.
- `void addLast(T& name)` – dodaje nowy element zawierający `name` na koniec listy.
- `void addLast(node<T>* n)` – dodaje element `n` na koniec listy.
- `node<T>* detachFront()` – odpina i zwraca pierwszy element listy. Jeśli lista jest pusta zwraca `NULL`.
- `node<T>* detachLast()` – odpina i zwraca ostatni element listy. Jeśli lista jest pusta zwraca `NULL`.
- `void reverse()` – odwraca aktualną listę.
- `void unionn(doubleList<T>& L)` – na koniec aktualnej listy dołącza elementy listy `L`. Po wykonaniu operacji lista `L` jest pusta.
- `void clean()` – usuwa wszystkie elementy listy.

Uwaga: Zadanie zrealizowane bez implementacji szablonu zostanie odrzucone przez prowadzącego.

Wejście

Pierwsza linia wejścia zawiera liczbę całkowitą z ($1 \leq z \leq 2 \cdot 10^9$) – liczbę zestawów danych, których opisy występują kolejno po sobie. Opis jednego zestawu jest następujący:

Pierwsza linia zawiera liczbę całkowitą n ($1 \leq n \leq 1000000$) będącą liczbą wypowiedzi Pana Macieja. W kolejnych n liniach znajdują się wypowiedzi. Wszystkie nazwy występujące w wypowiedziach składają się z dużych i małych liter alfabetu angielskiego i nie przekraczają 8 znaków. Do przechowywania nazw można zastosować klasę `string`.

Wyjście

Dla każdego zestawu danych wypisz kolejno opisy pociągów w reakcji na wypowiedzi `PRINT` z zestawu.

Dostępna pamięć: 64MB

Przykład

Dla danych wejściowych:

```
1
19
NEW Jeden Przemek
BACK Jeden Andrzej
FRONT Jeden Grzegorz
PRINT Jeden
NEW Dwa Adam
BACK Dwa Jan
FRONT Dwa Kamil
REVERSE Dwa
PRINT Dwa
NEW Trzy Mikolaj
FRONT Trzy Monika
PRINT Trzy
DELFONT Cztery Trzy
BACK Cztery Piotrek
PRINT Cztery
UNION Trzy Cztery
PRINT Trzy
DELBACK Trzy Cztery
PRINT Cztery
```

Poprawną odpowiedzią jest:

```
"Jeden":
Grzegorz<-Przemek<-Andrzej
"Dwa":
Jan<-Adam<-Kamil
"Trzy":
Monika<-Mikolaj
"Cztery":
Monika<-Piotrek
"Trzy":
Mikolaj<-Monika<-Piotrek
"Cztery":
Piotrek
```