Opis

Na Wyspie Sodor trwał gorący i słoneczny dzień. Do stacji przyjechały pociągi, by pożegnać swojego przyjaciela Tomka, który po długoletniej pracy odchodzi na emeryturę. Z tej okazji Gruby Zawiadowca zorganizował zabawę, w której brały udział wszystkie pociągi. W czasie zabawy Gruby Zawiadowca podawał polecenia, które mieli wykonywać zaproszeni goście.

Twoim zadaniem jest przeprowadzić symulacje zabawy używając ściśle określonych struktur danych do reprezentacji obiektów:

- Listy podwójnej cyklicznej z głową do reprezentacji pojedynczego pociągu,
- Listy pojedynczej bez głowy do reprezentacji zbioru pociągów.

Każdy pociąg ma swoja nazwę i co najmniej jeden wagon. Źródłem informacji jest poniższa lista poleceń Grubego Zawiadowcy dotycząca imprezy:

a. **`New T1 W`** – tworzy nowy pociąg o nazwie **`T1`** z jednym wagonem o nazwie **`W`** wstawia go do listy pociągów.

b. **`InsertFirst T1 W`** – wstawia wagon o nazwie **`W`** na początek pociągu o nazwie **`T1`**

c. **`InsertLast T1 W`** – wstawia wagon o nazwie **`W`** na koniec pociągu o nazwie **`T1`**

d. **`Display T1`** – wypisuje opis pociągu o nazwie **`T1`**

e. **`Trains`** – wypisuje aktualną listę pociągów, biorących udział w zabawie.

f. **`Reverse T1`** – odwraca kolejność wagonów w pociągu o nazwie **`T1`**

g. **`Union T1 T2`** – dołącza pociąg o nazwie o nazwie **`T2`** na koniec pociągu o nazwie **`T1`** i usuwa pociąg **`T2`** z listy pociągów

h. **`DelFirst T1 T2`** – usuwa pierwszy wagon z pociągu o nazwie **`T1`** i tworzy z niego nowy pociąg o nazwie **`T2`** i jeśli to był jedyny wagon w **`T1`** to **`T1`** przestaje istnieć (jest usuwany z listy pociągów).

i. **`DelLast T1 T2`** – usuwa ostatni wagon z pociągu o nazwie **`T1`** i tworzy z niego nowy pociągu o nazwie **`T2`**, przy czym, jeśli to był jedyny wagon w **`T1`** to **`T1`** przestaje istnieć (jest usuwany z listy pociągów).

Wejście

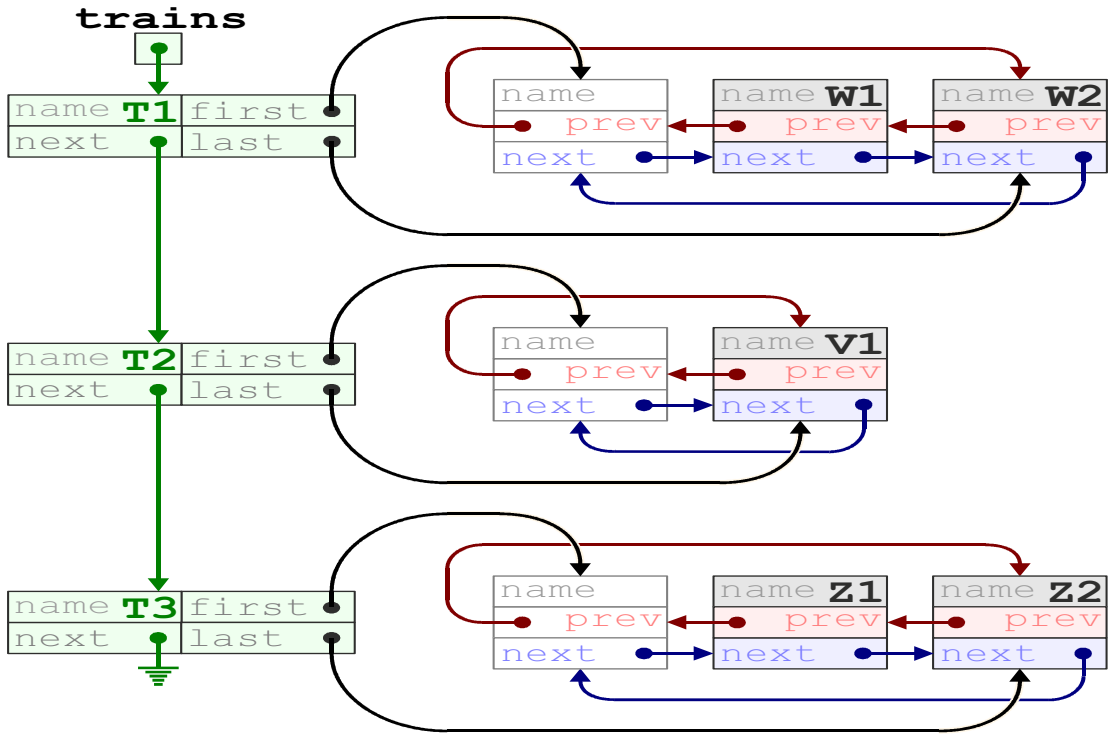
Pierwsza linia wejścia zawiera liczbę całkowitą ***z*** – liczbę zestawów danych, których opisy występują kolejno po sobie.

Pierwsza linia każdego zestawu zawiera liczbę całkowitą ***n*** ($1 \leq n \leq 10^6$) będącą liczbą poleceń, zaś każde polecenie umieszczone jest w osobnej linii i zawiera od jednego do trzech słów.

Pierwsze słowo jest nazwą polecenia i jest zawsze zakończone spacją, zaś pozostałe słowa, jeśli występują są jego parametrami, oddzielonymi pojedynczą spacją.

Nazwy pociągów i wagonów spełniają wymogi identyfikatorów stosowanych w programowaniu w języku Java, zaś nazwy poleceń są traktowane jako słowa zastrzeżone.

Przykładową listę trzech pociągów ilustruje poniższy rysunek:



Wyjście

- W reakcji na polecenia **`Display`** znajdujące się na wejściu wypisz kolejno opisy pociągów. Opis pociągu rozpoczyna się jego nazwą, zakończoną znakiem ':' i spacją, po której występują nazwy wagonów rozdzielanych znakiem spacji w kolejności od pierwszego do ostatniego wagonu.
- W reakcji na polecenia **Trains** znajdujące się na wejściu wypisz aktualną listę pociągów. Opis listy rozpoczyna się słowem **Trains**, zakończonym znakiem ':' i spacją, następnie występują nazwy pociągów rozdzielanych znakiem spacji w kolejności od pierwszego do ostatniego pociągu na liście.

Wymagania implementacyjne

1. Jedynym możliwym importem jest **java.util.Scanner.**
2. W szczególności zabronione są zarówno w całości jak i w jakiejkolwiek części importy **java.util.AbstractList** oraz **java.awt.List**.
3. **Wszystkie wymienione polecenia, poza *Display* oraz *Trains* muszą działać w czasie *O*(1) nie licząc pomocniczych operacji związanych z wyszukiwaniem zadanego pociągu i używać jak najmniej pamięci.**
4. **Wszystkie pomocnicze operacje jak np. wstawianie nowego pociągu, wyszukiwanie zadanego pociągu lub usuwanie pociągu zaimplementuj tak, aby zawierały minimalną liczbę przeglądów list.**

5. **Program powinien sprawdzać czy wszystkie polecenia są sensowne np.**
   (a) **czy nie utworzą drugiego pociągu o tej samej nazwie, w przeciwnym przypadku program wypisuje komunikat: Train *name* already exists**
   (b) **czy nie zostaną użyte do łączenia, odwracania wypisywania, wstawiania lub usuwania wagonów nieistniejących pociągów, w przeciwnym przypadku wypisuje komunikaty: Train *name* does not exist**

Przykład danych

Wejście:	**Wynik:**
1 21 New T1 W1 InsertLast T1 W2 Display T1 New T1 W0 InsertFirst T1 W0 Display T1 Display T5 DelFirst T1 T2 Display T1 Display T2 DelLast T1 T3 Display T1 Display T3 DelLast T6 T1 DelFirst T1 T2 Union T9 T8 Reverse T1 Display T1 New T9 Z0 Display T9 Trains	T1: W1 W2 Train T1 already exists T1: W0 W1 W2 Train T5 does not exist T1: W1 W2 T2: W0 T1: W1 T3: W2 Train T6 does not exist Train T2 already exists Train T9 does not exist Train T1 does not exist Train T1 does not exist T9: Z0 Trains: T9 T3 T2