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 bez głowy do reprezentacji wagonów pojedynczego pociągu,
- Listy pojedynczej bez głowy do reprezentacji zbioru pociągów.

Każdy pociąg składa się z lokomotywy i co najmniej jednego wagonu. Możemy przyjąć, że nazwą pociągu jest nazwa lokomotywy.

Gruby Zawiadowca przygotował poniższą listę poleceń dotyczącą imprezy:

a. **New T1 W** – tworzy nowy pociąg zawierający lokomotywę o nazwie **T1** z jednym wagonem o nazwie **W** i 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. **TrainsList** – 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

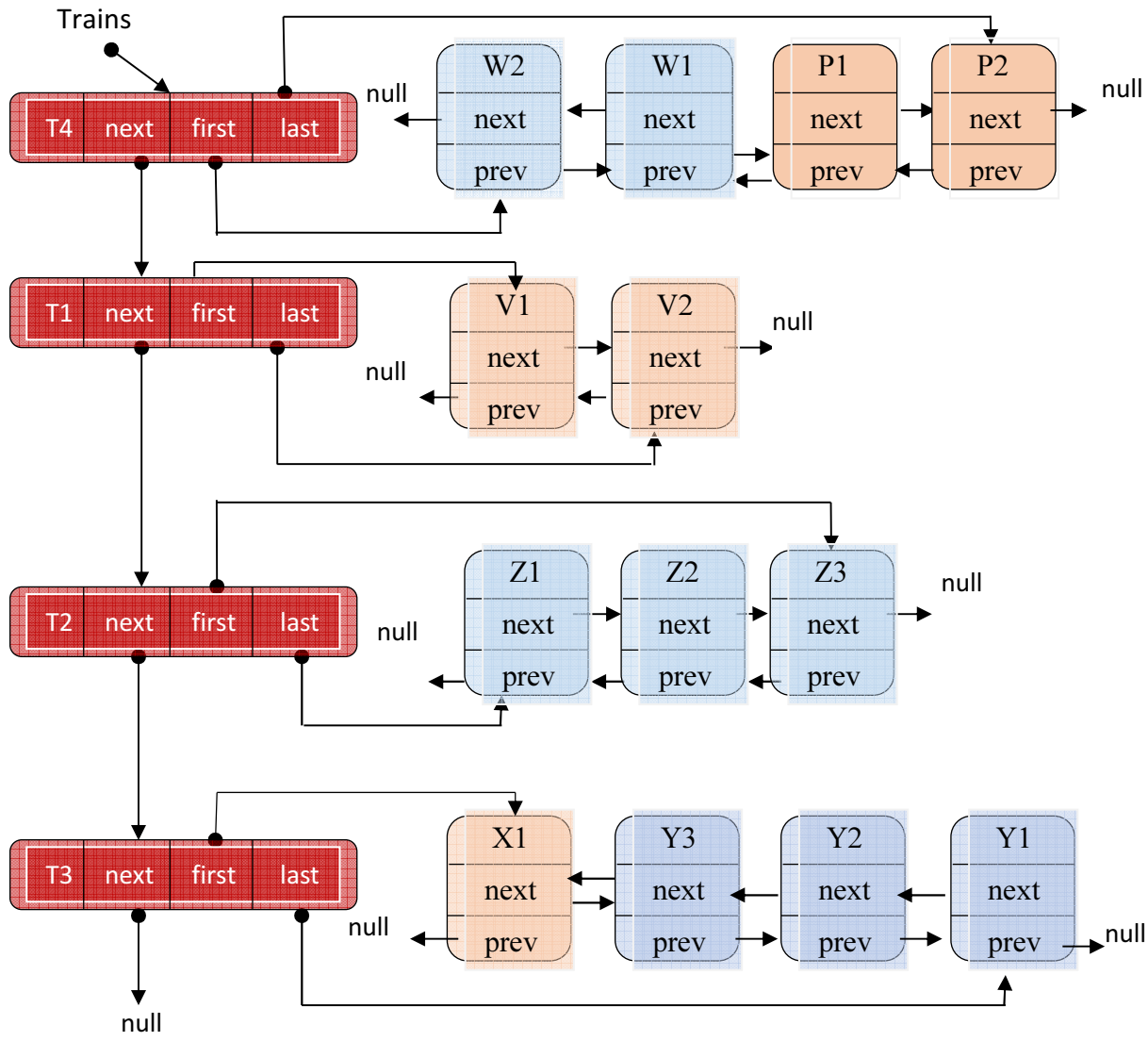
Pierwsza linia wejścia zawiera liczbę całkowitą ***z*** – liczbę zestawów danych, których opisy występują kolejno po sobie.

Pierwsza linia każdego zestawu zawiera liczbę całkowitą ***n*** ($1 <= n <= 10^6$) 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 czterech pociągów ilustruje poniższy rysunek:



Wyjście

- W reakcji na polecenia: **`Display nazwa_pociągu`** wypisz aktualną listę wagonów zadanego pociągu. Opis listy rozpoczyna się nazwą pociągu, zakończoną znakiem ‘:’ i spacją, po której występują nazwy wagonów rozdzielanych znakiem spacji w kolejności od pierwszego do ostatniego wagonu na liście.
- W reakcji na polecenia **TrainsList** 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.
- Na końcu każdej z list znajduje znak nowej linii.

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 *TrainsList* muszą działać w czasie $O(1)$ 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. Możesz założyć, ze wszystkie polecenia są sensowne, to znaczy nie utworzą drugiego pociągu o tej samej nazwie, ani też nie zostaną użyte do łączenia czy odwracania nieistniejącego pociągu.

Przykład danych

Wejście:	Wynik:
1	T1: W1 W2
23	T1: W0 W1 W2
New T1 W1	T1: W1 W2
InsertLast T1 W2	T2: W0
Display T1	T1: W1
InsertFirst T1 W0	T3: W2
Display T1	Trains: T3 T2 T1
DelFirst T1 T2	T4: Z2 Z1
Display T1	T3: W2 Z2 Z1
Display T2	Trains: T3 T2 T1
DelLast T1 T3	T3: W2 Z2 Z1 W0
Display T1	T3: W0 Z1 Z2 W2
Display T3	
TrainsList	
New T4 Z1	
InsertLast T4 Z2	
Reverse T4	
Display T4	
Union T3 T4	
Display T3	
TrainsList	
Union T3 T2	
Display T3	
Reverse T3	
Display T3	