Opis

Na Wyspie Sodor trwał gorący i słoneczny dzień. Na zaproszenie Grubego Zawiadowcy do stacji przyjechały pociągi, by wziąć udział w zabawie na cześć pociągu o nazwie Tomek, który po długoletniej pracy odchodzi na emeryturę. Każdy pociąg ma swoją nazwę i *listę wagonów*, zawierającą co najmniej jeden wagon. *Lista pociągów* biorących udział w zabawie nie zawiera duplikatów.

W czasie zabawy obowiązki wodzireja pełnił Gruby Zawiadowca, który podawał poniższe polecenia:

a. *New T1 W* – tworzy nowy pociąg 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 listę wagonów pociągu o nazwie *T1* począwszy od pierwszego wagonu

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).

Twoim zadaniem jest przeprowadzić symulację zabawy używając efektywnej implementacji poniższych struktur danych:

- jednostronnej listy dwukierunkowej, cyklicznej, której referencja *first* wskazuje pierwszy element listy dla reprezentacji *listy wagonów* danego pociągu.
- listy pojedynczej, której refencja *trains* wskazuje pierwszy element listy dla reprezentacji *listy pociągów*.

Wejście

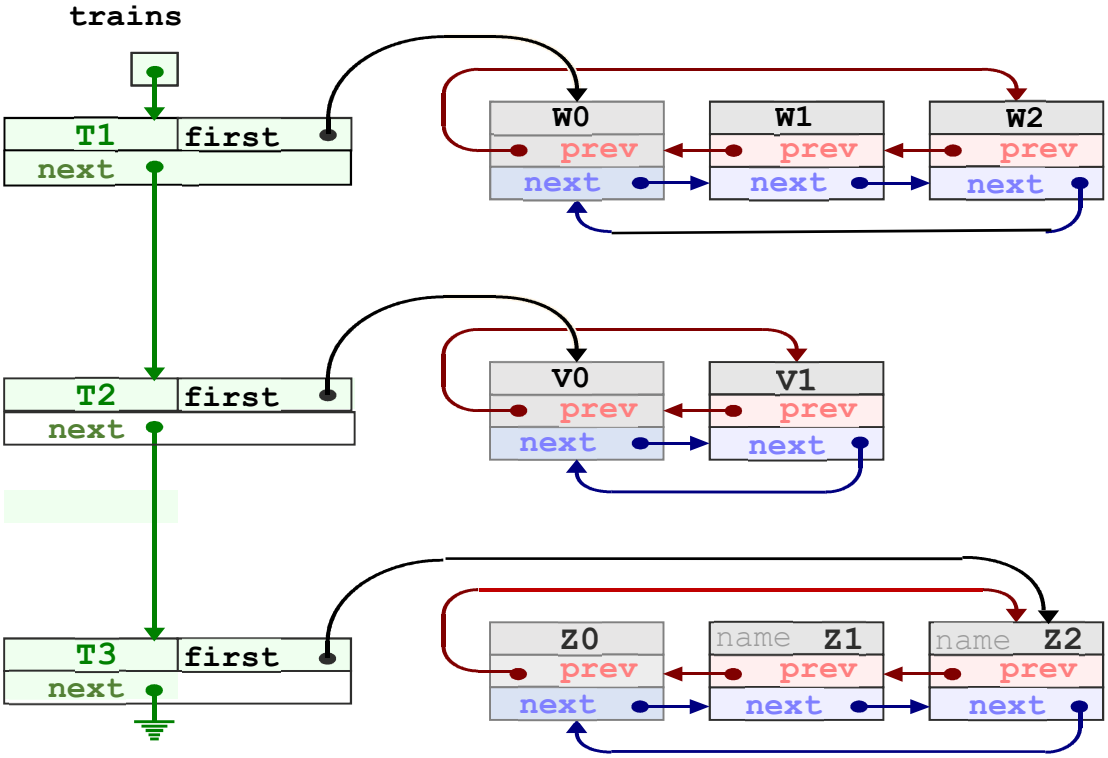
Pierwsza linia wejścia zawiera liczbę całkowitą ***z*** – liczbę zestawów danych, których opisy występują kolejno po sobie.

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. Każde polecenie kończy się znakiem nowej linii.

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 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.
- W obu przypadkach wyświetlane listy kończą się znakiem nowej linii.

Wymagania implementacyjne

- Jedynym możliwym importem jest **java.util.Scanner**. W szczególności zabronione są zarówno w całości jak i w jakiejkolwiek części importy **java.util.AbstractList** oraz **java.awt.List**.
- Deklaracje klas *lista pociągów* i *lista wagonów* muszą być zgodne z podanym wyżej opisem, w przeciwnym przypadku program zostanie odrzucony pomimo akceptacji przez BaCę.

- Wszystkie wymienione polecenia, poza *Display* oraz *Trains* muszą używać jak najmniej pamięci i działać w czasie *O(1)* nie licząc pomocniczych operacji związanych z wyszukiwaniem zadanego pociągu.
- Wszystkie pomocnicze operacje jak np. wstawianie nowego pociągu, wyszukiwanie zadanego pociągu lub usuwanie pociągu zaimplementuj tak, aby zawierały minimalną liczbę przeglądów list.
- Elementy *listy pociągów* nie mogą mieć dodatkowych pól postaci *boolean reserwed*.
- Program powinien sprawdzać czy wszystkie polecenia są sensowne np.
    a. nie tworzą duplikatów pociągów, w przeciwnym przypadku program wypisuje komunikat: *Train name already exists*
    b. nie odwołują się do pociągów - nieistniejących na liście, w przeciwnym przypadku wypisuje komunikaty: *Train name does not exist*
- W przypadku operacji *DelFirst(T1, T2)* lub *DelLast(T1,T2),* program działa w tzw. "krótkim obiegu". To znaczy wykonanie usuń jest realizowane tylko gdy istnieje pociąg *T1* i nie istnieje *T2*.

Przykład danych

Wejście:	Wynik:
1	T1: W1
14	Train T1 already exists
New T1 W1	T1: W1
Display T1	T1: W1 W2
New T1 W0	T1: W1
Display T1	T2: W2
InsertLast T1 W2	T3: W1
Display T1	T2: W2
DelLast T1 T2	Train T1 does not exist
Display T1	Trains: T3 T2
Display T2	
DelFirst T1 T3	
Display T3	
Display T2	
Display T1	
Trains	