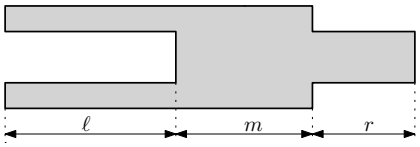
A. Chodnik

Dostępna pamięć: 16 MB

Do układania chodnika dostarczono kostki w kształcie przedstawionym na poniższym rysunku. Każda taka kostka jest charakteryzowana przez trzy liczby ℓ, m i r będące długościami lewego łącznika, środka i prawego łącznika.



Twoim zadaniem jest wybranie podzbioru kostek i połączenie ich w chodnik, tak żeby:

1. chodnik dobrze się zaczynał: długość lewego łącznika pierwszej kostki wynosiła zero;
2. chodnik był dobrze połączony: dla dwóch kolejnych kostek długość prawego łącznika pierwszej z nich była równa długości lewego łącznika drugiej z nich;
3. chodnik dobrze się kończył: długość prawego łącznika ostatniej kostki wynosiła zero.

Każdą kostkę można wykorzystać tylko raz. Kostek nie wolno obracać (zresztą i tak by wtedy nie pasowały).

Specyfikacja danych wejściowych

W pierwszym wierszu wejścia znajduje się liczba naturalna $n \in [1, 200\,000]$ będąca liczbą dostępnych kostek. W każdym z kolejnych n wierszy znajduje się opis kolejnej kostki, będący trzema liczbami naturalnymi ℓ, m i r charakteryzującymi kostkę. Liczby te spełniają warunki $\ell, r \in [0, 10\,000]$ oraz $m \in [1, 10\,000]$. Każde dwie kostki są różne.

Specyfikacja danych wyjściowych

Jeśli ułożenie kostek w chodnik nie jest możliwe, w pierwszym i jedynym wierszu wyjścia Twój program powinien wypisać słowo `BRAK`. W przeciwnym przypadku w pierwszym wierszu Twój program powinien wypisać liczbę całkowitą dodatnią s będącą liczbą kostek wchodzącą w skład chodnika. Następnie w każdym z kolejnych s wierszy powinien znaleźć się opis kolejnej kostki wchodzącej w skład chodnika (trzy liczby, tak jak w danych wejściowych). Jeśli istnieje więcej niż jedno poprawne rozwiązanie, Twój program może wypisać dowolne z nich.

Przykład A

Wejście:
```
6
0 1 1
1 1 3
3 1 2
3 1 0
2 1 4
2 1 3
```

Wyjście:
```
3
0 1 1
1 1 3
3 1 0
```

Przykład B

Wejście:
```
5
0 1 1
0 2 2
0 3 2
2 4 3
3 5 0
```

Wyjście:
```
3
0 2 2
2 4 3
3 5 0
```

Przykład C

Wejście:
```
1
0 1 2
```

Wyjście:
```
BRAK
```