IOI 2002

Specchio (`specchio`)

Descrizione del problema

Un *albero (ordinato)* è formato da una *radice* e da un insieme finito e ordinato (eventualmente vuoto) di *figli*, che sono anch'essi alberi. Ad esempio, la Figura 1 rappresenta un albero: per convenzione, la radice viene disegnata in cima (cioè al contrario rispetto agli alberi veri). Come ogni albero, lo potete descrivere dicendo quanti figli ha la radice (in questo caso, 4) e poi descrivendo uno a uno, nell'ordine da sinistra a destra, i 4 sotto-alberi. In questo modo, ad esempio, l'albero in Figura 1 sarebbe identificato dalla seguente sequenza:

4 2 0 3 0 0 1 0 0 0 0

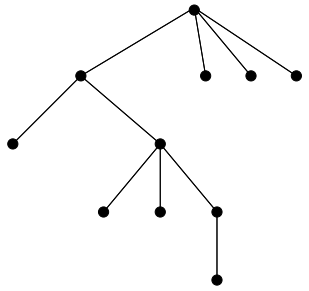

Figura 1: Un albero.

Infatti, l'albero ha quattro figli sotto la radice, per cui il primo numero è 4. A questo 4 segue poi subito la sequenza 2 0 3 0 0 1 0, che è la descrizione del primo figlio, mentre gli ultimi tre figli sono ciascuno descritti da 0 (dato che non hanno figli).

Supponete ora di guardare l'albero riflesso in uno specchio. L'ordine dei figli di ciascun nodo risulta invertito e l'albero di Figura 1 appare come in Figura 2.

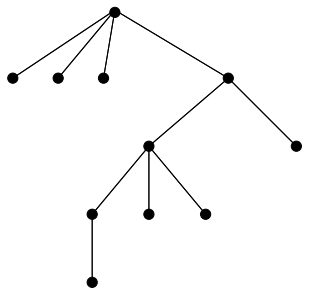

Figura 2: L'albero di Figura 1 allo specchio.

Questo nuovo albero è descritto dalla sequenza

4 0 0 0 2 3 1 0 0 0 0

Fornire un metodo per passare dalla descrizione di un albero alla descrizione del suo rovesciato sinistra-destra (e viceversa).

Dati di input

Il vostro programma legge da `stdin` una sola riga contenente una sequenza di interi non negativi, separati da spazi. La sequenza descrive correttamente un albero: quindi il primo numero è il numero di figli della radice, ed è seguito dalle descrizioni dei sotto-alberi radicati nei figli della radice, prese una dopo l'altra, nel loro ordine da sinistra a destra.

Dati di output

Il vostro programma deve restituire su `stdout` una sola riga, costituita da una sequenza di interi non negativi separati da spazi. Tale sequenza è intesa codificare, secondo le stesse regole che per l'input già viste sopra, l'albero rovesciato sinistra-destra, ossia l'albero dato in input visto allo specchio.

Esempio di input/output

input (da stdin)	output (su stdout)
4 2 0 3 0 0 1 0 0 0 0	4 0 0 0 2 3 1 0 0 0 0

input (da stdin)	output (su stdout)
4 0 0 0 2 3 1 0 0 0 0	4 2 0 3 0 0 1 0 0 0 0

Subtask

[2 istanze] esempi_testo: i due esempi del testo.

[20 istanze] small: $N \leq 10$.

[20 istanze] medium: $N \leq 100$.

[14 istanze] huge_binary: $N \leq 10\,000$.

[14 istanze] huge_max_2_children: $N \leq 100\,000$.

[20 istanze] huge: $N \leq 100\,000$.

In generale, quando si richiede la valutazione di un subtask vengono valutati anche i subtask che li precedono, ma si evita di avventurarsi in subtask successivi fuori dalla portata del tuo programma che potrebbe andare in crash o comportare tempi lunghi per ottenere la valutazione completa della sottomissione. Ad esempio, chiamando:

```
rtal -s wss://ta.di.univr.it/algo  connect -a size=big  specchio -- python my_solution.py
```

vengono valutati, nell'ordine, i subtask:

`esempi_testo`, `small`, `medium`, `big`.

Il valore di default per l'argomento `size` è `huge`.

Il tempo limite per istanza (ossia per ciascun testcase) è sempre di 1 secondo.