Biscotti dal Sapore Unico

Nome del problema	Twin Cookies
File di input	standard input
File di output	standard output
Limite di tempo	1 secondo
Limite di memoria	256 megabyte

Questo è un problema interattivo. Il tuo programma comunicherà con il grader alternando la scrittura di messaggi nello standard output con la lettura di messaggi dallo standard input.

Sophie sta organizzando la festa di compleanno dei suoi gemelli, che adorano i biscotti. Per il loro compleanno vorrebbero qualcosa di nuovo: biscotti preparati dalla BSU (*Biscotti dal Sapore Unico*).

Ogni biscotto prodotto dalla BSU ha un valore intero di bontà tra 1 e 10^{16} inclusi. Per non fare differenze tra i gemelli, Sophie vuole fare in modo che la somma delle bontà dei biscotti ricevuti da ciascuno di loro sia la stessa.

La BSU accetta solo ordini di **esattamente** n biscotti. In ogni ordine il cliente deve specificare i valori di bontà di ciascuno degli n biscotti richiesti.

Fedele al suo nome, la *Biscotti dal Sapore Unico* si rifiuta di produrre, per lo stesso cliente, due biscotti di bontà uguale. Sophie deve quindi assicurarsi di non ordinare mai la stessa bontà due volte, né nello stesso ordine, né in ordini diversi. È la prima volta che Sophie ordina dalla BSU, quindi inizialmente ha a disposizione tutti i possibili valori di bontà.

La BSU ha la brutta fama di offrire un servizio di spedizione pessimo. Ogni volta che un cliente ordina n biscotti, solo uno di questi arriva a destinazione (gli altri vengono mangiati dal corriere!). Il cliente non può sapere in anticipo quale di questi biscotti riceverà.

Il compleanno si sta avvicinando, e Sophie ha tempo per fare al più 101 ordini.

Aiutala:

1. Per prima cosa, ordina i biscotti. Puoi fare al più 101 ordini, uno alla volta, e

ciascuno con esattamente n valori di bontà. **Immediatamente dopo ogni ordine vieni messo al corrente della bontà dell'unico biscotto ricevuto**.

Ricorda che non puoi ordinare la stessa bontà più di una volta, anche tra ordini diversi. In particolare, se ordini un biscotto di bontà b ma questo non arriva, **non puoi** ordinare un altro biscotto di bontà b.

2. In seguito, dividi i biscotti. Quando hai ricevuto abbastanza biscotti, dovrai distribuire **alcuni** di essi tra i gemelli. Ogni gemello deve ricevere almeno un biscotto e la somma delle bontà ricevute deve coincidere. **Puoi anche non usare tutti i biscotti ricevuti!**

Output

Ogni volta che il tuo programma scrive una o più righe nello standard output, ti devi ricordare di ***flushare* lo standard output**.

Ciò può essere fatto, ad esempio, nei seguenti modi:

- In C++, ci sono diverse opzioni:
  - `fflush(stdout);`
  - `std::cout << std::flush;`
  - `std::cout << std::endl;` (nota che questo stamperà anche un a capo)
  - anche leggere da `std::cin` *flusha* l'output
- in Python, puoi usare `sys.stdout.flush()`

Modalità di interazione

Il tuo programma deve eseguire questa sequenza di operazioni:

1. Leggere il valore di n dallo stardard input.
2. Al più 101 volte:
   1. Scrivi nello stardard output una riga contenente `?` seguito da uno spazio e dai valori di bontà degli n biscotti da ordinare (separati da uno spazio).
   2. Poi, leggi dallo standard input il valore di bontà dell'unico biscotto ricevuto. È garantito che questo è uno degli n valori dell'ultimo ordine.
3. Stampa tre righe che descrivono quali biscotti ricevono i gemelli:
   1. La prima riga ha la forma "`!` m k", dove $m, k > 0$ sono le quantità di biscotti distribuiti a ciascun gemello.
   2. La seconda riga contiene m interi separati da spazi: i valori di bontà assegnati al primo gemello.
   3. La terza riga contiene k interi separati da spazi: i valori di bontà assegnati al secondo gemello.

L'output deve soddisfare le seguenti condizioni:

1. Ogni gemello deve ricevere almeno un biscotto.
2. I gemelli devono ricevere biscotti con la stessa bontà totale.
3. Puoi usare solo biscotti effettivamente ricevuti.
4. Puoi dare ciascun biscotto ricevuto ad al più un gemello.

Puoi stampare i valori di bontà in un qualsiasi ordine.

Dopo aver finito *flusha* lo standard output un'ultima volta e **termina normalmente il programma**.

Assegnazione del punteggio

Subtask 1 (8 punti): $n = 1$

Subtask 2 (9 punti): $1 \leq n \leq 2$

Subtask 3 (18 punti): $1 \leq n \leq 25$

Subtask 4 (16 punti): $1 \leq n \leq 200$

Subtask 5 (13 punti): $1 \leq n \leq 1000$

Subtask 6 (36 punti): $1 \leq n \leq 5000$

Esempi

standard input	standard output
1 13 7 31 12 5 3	? 13 ? 7 ? 31 ? 12 ? 5 ? 3 ! 2 3 7 13 12 5 3
2 7 2 5	? 3 7 ? 2 8 ? 1 5 ! 2 1 2 5 7

Note

Gli esempi vanno letti alternando una riga dell'input con una riga dell'output.

Il grader è adattivo, cioè non sceglie necessariamente in modo casuale quale biscotto arriva a destinazione.

Per $n = 2$, facendo la stessa sequenza di ordini del secondo esempio, potresti ricevere biscotti diversi.