Kekse für Zwillinge

Name der Aufgabe	Twin Cookies
Eingabe	standard input
Ausgabe	standard output
Zeitlimit	1 Sekunde
Speicherlimit	256 MB

Dies ist eine interaktive Aufgabe. Dein Programm wird mit unserem Grader kommunizieren, indem es abwechselnd Nachrichten in die Ausgabe schreibt und Nachrichten von der Eingabe liest.

Sophie trifft Vorbereitungen für die Geburtstagsparty ihrer Zwillingstöchter. Die Zwillinge lieben Kekse! Für ihren Geburtstag würden sie daher gern etwas Neues ausprobieren: Kekse von der Unique Cookie Tastiness Company (UCTC).

Jeder UCTC-Keks hat einen ganzzahligen Geschmackswert von 1 bis 10^{16} (inklusive). Die Summen der Geschmackswerte der Kekse, welche die Zwillingstöchter jeweils erhalten, müssen exakt gleich sein, weil sonst natürlich eine von ihnen eifersüchtig würde.

Die UCTC akzeptiert nur Bestellungen von **genau** n Keksen. In jeder Bestellung muss der Käufer angeben, welche Geschmackswerte die n bestellten Kekse haben sollen.

Um ihrem Namen gerecht zu werden, weigert sich die Unique Cookie Tastiness Company, zwei oder mehr Kekse des gleichen Geschmackswerts für den gleichen Käufer zu produzieren. Also muss Sophie darauf achten, nicht zweimal den gleichen Geschmackswert zu bestellen – weder in einer Bestellung, noch in verschiedenen Bestellungen. Sophie hat zuvor noch nie bei UCTC bestellt; daher kann sie jeden Geschmackswert genau einmal bestellen.

Es gibt allerdings noch ein weiteres Problem: Bekanntlich ist der Lieferservice von UCTC grauenhaft. Wenn ein Käufer n Kekse bestellt, kommt nur einer dieser n Kekse tatsächlich beim Käufer an. Der Rest wird auf dem Weg von Angestellten des Lieferservices gegessen. Der Käufer kann leider nicht beeinflussen, welcher der n Kekse tatsächlich bei ihm ankommt.

Der Geburtstag naht, deshalb hat Sophie gerade noch Zeit für höchstens 101

Bestellungen! Deine Aufgabe ist es, ihr zu helfen.

Genauer gesagt sollst du Folgendes tun:

1. Zuerst bestellst du die Kekse. Du darfst höchstens 101 Bestellungen machen, jede davon bestehend aus genau n gewünschten Geschmackswerten. Dabei machst du immer eine Bestellung auf einmal. **Direkt nach jeder Bestellung erfährst du, welcher Keks tatsächlich bei dir angekommen ist.**

   Denk daran, dass du niemals den gleichen Geschmackswert bestellen darfst, nicht einmal in verschiedenen Bestellungen. (Insbesondere darfst du einen Geschmackswert **nicht** nochmal bestellen, auch wenn der Keks beim ersten Mal nicht angekommen ist.)

2. Dann verteilst du die Kekse. Sobald du genug Kekse erhalten hast, solltest du **einige** der erhaltenen Kekse zwischen den Zwillingsschwestern aufteilen. Jede muss natürlich mindestens einen Keks erhalten, und die Summen der Geschmackswerte der Kekse, die die Zwillingstöchter jeweils erhalten, muss exakt gleich sein. **Du musst dabei nicht alle Kekse, die du erhalten hast, benutzen.**

Ausgabe

Jedes Mal, wenn du etwas ausgibst, musst du den **Outputstream flushen**. Das ist nötig, damit deine Ausgabe unseren Grader sofort erreicht.

Das geht zum Beispiel so:

- In C++ gibt es verschiedene Möglichkeiten.
  - `fflush(stdout);`
  - `std::cout << std::flush;`
  - `std::cout << std::endl;` (beachte, dass das zusätzlich einen Zeilenumbruch ausgibt!)
  - `std::cin` flusht auch die Ausgabe.
- In Java kannst du `System.out.flush()` verwenden.
- In Python kannst du `sys.stdout.flush()` verwenden.

Interaktion

Dein Programm sollte Folgendes tun:

1. Die Ganzzahl n aus der Standardeingabe lesen.
2. Höchstens 101 mal:
   1. Zuerst gibst du eine Zeile aus, die eine Bestellung von n Keksen beschreibt.
   2. Anschließend liest du den Geschmackswert des Kekses, den du erhalten hast, von der Eingabe. Es ist garantiert, dass dies einer der

Geschmackswerte ist, die du unmittelbar zuvor bestellt hast.
3. Gib drei Zeilen aus, die eine gültige Möglichkeit beschreiben, die Kekse aufzuteilen.

Der Grader wird jede Ganzzahl in eine eigene Zeile schreiben.

Um Kekse zu bestellen, gib eine einzelne Zeile mit ? aus, gefolgt von n durch einzelne Leerzeichen getrennten Ganzzahlen: den Geschmackswerten der Kekse, die du bestellen möchtest.

Denk daran, dass du höchstens 101 Bestellungen machen darfst, und dass du den gleichen Geschmackswert nicht mehrfach verwenden kannst.

Sobald du genug Kekse bestellt hast, gib die letzten drei Zeilen aus, die beschreiben, welche Kekse Sophie ihren Zwillingstöchtern geben soll.

Die erste dieser Zeilen sollte die Form "! m k" mit $m, k > 0$ haben: die Anzahlen m und k der Kekse, die die erste und zweite Tochter jeweils erhalten sollen, in dieser Reihenfolge.

Die zweite dieser Zeilen sollte m Ganzzahlen, die durch einzelne Leerzeichen getrennt sind, enthalten: die Geschmackswerte der Kekse, die die erste Zwillingstochter erhalten soll.

Analog dazu soll die dritte dieser Zeilen k durch einzelne Leerzeichen getrennte Ganzzahlen enthalten: die Geschmackswerte der Kekse, die die zweite Zwillingstocher erhalten soll.

Die Ausgabe muss die folgenden Bedingungen erfüllen:

1. Jede Zwillingstochter muss mindestens einen Keks erhalten.
2. Jede Zwillingstochter muss Kekse mit der gleichen Summe der Geschmackswerte erhalten.
3. Nur Kekse, die nach den Bestellungen tatsächlich angekommen sind, dürfen verwendet werden.
4. Jeder Keks kann natürlich nur einer der Töchter gegeben werden.

Jede Ausgabe, die diese Bedingungen erfüllt, wird akzeptiert. Insbesondere kannst du die Kekse in beliebiger Reihenfolge ausgeben.

Nachdem du die drei letzten Zeilen ausgegeben hast, musst du ein letztes Mal den Outputstream flushen und dann dein **Programm normal beenden**.

Subtasks

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

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

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

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

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

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

Beispiele

Eingabe	Ausgabe
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

Die Beispiele von Ein- und Ausgabe sollten Zeile für Zeile gelesen werden. Das Programm liest abwechselnd einen Wert von der Eingabe und schreibt eine Zeile (bzw. drei Zeilen am Ende) in die Ausgabe.

Der Grader wählt willkürlich, welcher Keks geliefert wird. Das bedeutet, der Grader kann sich in manchen Testfällen an deine Ausgaben anpassen, und in manchen Testfällen einfach zufällig einen Keks wählen. Insbesondere könnte er für $n = 2$, selbst wenn du die gleichen Bestellungen wie im Beispiel machst, eine andere Menge an Keksen zurückgeben.