Сувенири

Амару купува сувенири в чуждестранен магазин. Съществуват N **вида** сувенири. В магазина се предлагат безкраен брой сувенири от всеки вид.

Всеки вид сувенир има фиксирана цена. Сувенир от вид i ($0 \leq i < N$) струва $P[i]$ монети, където $P[i]$ е положително цяло число.

Амару знае, че видовете сувенири са подредени в низходящ ред по цена и че цените на сувенирите са различни. По-конкретно, $P[0] > P[1] > \cdots > P[N-1] > 0$. Още повече, той е успял да научи колко струва първият вид сувенир, тоест стойността $P[0]$. За съжаление, Амару няма допълнителна информация за цените на сувенирите.

За да купи сувенири, Амару ще извърши редица транзакции с продавача.

Всяка транзакция се състои от следните стъпки:

1. Амару подава някакъв (положителен) брой монети на продавача.
2. Продавачът слага тези монети върху маса в задната част на магазина, където Амару не може да ги види.
3. Продавачът преминава през видовете сувенири $0, 1, \ldots, N-1$ в този ред, един по един. Всеки вид бива разгледан **точно един път** на транзакция.
    - Когато разглежда вид i, ако сегашният брой монети на масата е поне $P[i]$,
        - продавачът премахва $P[i]$ монети от масата и
        - продавачът слага сувенир от вид i на масата.
4. Продавачът дава на Амару всички оставащи монети на масата и всички сувенири сложени на масата.

Обърнете внимание, че на масата няма монети или сувенири преди започването на транзакция.

Вашата задача е да инструктирате Амару да извърши определен брой транзакции, така че:

- във всяка транзакция той купува **поне един** сувенир и
- общо той купува **точно** i сувенира от вид i, за всяко i, $0 \leq i < N$. Забележете, че това означава че Амару трябва да не купи нито един сувенир от вид 0.

Не е необходимо Амару да минимизира броя транзакции. Също така, той разполага с безкраен брой монети.

Детайли по имплементацията

Трябва да имплементирате следната функция.

```
void buy_souvenirs(int N, long long P0)
```

- N: брой видове сувенири.
- $P0$: стойността $P[0]$.
- Функцията ще бъде извикана точно веднъж за всеки тест.

Горната функция може да прави извиквания към следната функция, за да инструктира Амару да осъществи транзакция:

```
std::pair<std::vector<int>, long long> transaction(long long M)
```

- M: броят монети дадени на продавача от Амару.
- Функцията връща pair. Първият елемент на pair-а е масив L, съдържащ видовете сувенири, които са купени (във възходящ ред). Вторият елемент е цяло число R, броят монети, върнати на Амару, след транзакцията.
    - Трябва да е изпълнено, че $P[0] > M \geq P[N-1]$. Условието $P[0] > M$ гарантира, че Амару няма да купи сувенир от вид 0, а $M \geq P[N-1]$ гарантира че Амару ще купи поне един сувенир. Ако тези условия не бъдат изпълнени, Вашето решение ще получи съобщение `Output isn't correct: Invalid argument`. Забележете, че за разлика от $P[0]$, стойността на $P[N-1]$ не е предоставена във входа.
- Функцията може да бъде извикана най-много 5000 пъти във всеки тест.

Грейдърът **не е адаптивен**. Това означава, че цените P са фиксирани преди функцията `buy_souvenirs` да бъде извикана.

Ограничения

- $2 \leq N \leq 100$
- $1 \leq P[i] \leq 10^{15}$ за всяко i, спазващо $0 \leq i < N$.
- $P[i] > P[i+1]$ за всяко i, спазващо $0 \leq i < N-1$.

Подзадачи

Подзадача	Точки	Допълнителни ограничения
1	4	$N = 2$
2	3	$P[i] = N - i$ за всяко i, спазващо $0 \leq i < N$.
3	14	$P[i] \leq P[i + 1] + 2$ за всяко i, спазващо $0 \leq i < N - 1$.
4	18	$N = 3$
5	28	$P[i + 1] + P[i + 2] \leq P[i]$ за всяко i, спазващо $0 \leq i < N - 2$. $P[i] \leq 2 \cdot P[i + 1]$ за всяко i, спазващо $0 \leq i < N - 1$.
6	33	Няма допълнителни ограничения.

Пример

Нека разгледаме следното извикване.

```
buy_souvenirs(3, 4)
```

Съществуват $N = 3$ вида сувенири и $P[0] = 4$. Забележете, че има само три възможни редици от цени P: $[4, 3, 2]$, $[4, 3, 1]$, и $[4, 2, 1]$.

Примете, че `buy_souvenirs` извиква `transaction(2)`. Нека функцията върне $([2], 1)$, което означава, че Амару е купил един сувенир от вид 2 и продавачът му е върнал 1 монета. Забележете, че това ни позволява да отгатнем, че $P = [4, 3, 1]$, защото:

- За $P = [4, 3, 2]$, `transaction(2)` би върнала $([2], 0)$.
- За $P = [4, 2, 1]$, `transaction(2)` би върнала $([1], 0)$.

Тогава `buy_souvenirs` може да викне `transaction(3)`, която връща $([1], 0)$, а това означава, че Амару е купил един сувенир от вид 1 и продавачът му е върнал 0 монети. Досега общо той е купил един сувенир от вид 1 и един сувенир от вид 2.

Най-накрая, `buy_souvenirs` може да викне `transaction(1)`, която връща $([2], 0)$, а това означава, че Амару е купил един сувенир от вид 2. Забележете, че бихме могли да използваме и `transaction(2)`. В този момент, Амару има общо един сувенир от вид 1 и два сувенира от вид 2, както се изисква.

Локален грейдър

Формат на входа:

```
N
P[0]  P[1] ... P[N-1]
```

Формат на изхода:

```
Q[0]  Q[1] ... Q[N-1]
```

Тук $Q[i]$ е общият броя купени сувенири от вид i, за всяко i, $0 \leq i < N$.