纪念品（souvenirs）

Amaru 在一家国外商店购买纪念品。商店有 N **种**纪念品，而且每种纪念品有无限多件现货。

每种纪念品都有一个固定的价格：第 i 种（$0 \leq i < N$）纪念品的价格为 $P[i]$ 枚硬币，其中 $P[i]$ 是一个正整数。

Amaru 知道纪念品种类是按价格降序排列的，而且所有纪念品的价格互不相同。具体来说，$P[0] > P[1] > \cdots > P[N-1] > 0$。此外，他还知道 $P[0]$ 的值。不幸的是，Amaru 没有其他纪念品的价格信息。

为了购买纪念品，Amaru 会与卖家进行若干次交易。

每次交易由以下步骤组成：

1. Amaru 向卖家支付一定数量（正数）的硬币。
2. 卖家将这些硬币堆放在商店后台的桌子上，因此 Amaru 无法看到这些硬币。
3. 卖家按顺序依次处理每种纪念品 $0, 1, \ldots, N-1$。每种纪念品在每次交易中被处理**恰好一次**。
    - 当处理第 i 种纪念品时，如果当前硬币堆中的硬币数量至少为 $P[i]$，则
        - 卖家从硬币堆中移除 $P[i]$ 枚硬币；
        - 卖家将一件第 i 种纪念品放在桌子上。
4. 卖家将剩余的所有硬币和桌子上的纪念品交给 Amaru。

注意，在每次交易开始前，桌上没有任何硬币或纪念品。

你的任务是指导 Amaru 进行若干次交易，使得：

- 在每次交易中，他购买**至少一件**纪念品；
- 总体上他**恰好**购买了 i 件第 i 种纪念品，其中 $0 \leq i < N$。注意，这意味着 Amaru 不应购买第 0 种纪念品。

Amaru 不需要最小化交易次数，而且拥有无限量的硬币供应。

实现细节

你需要实现以下函数。

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

- N：纪念品的种数。

- $P0$：$P[0]$ 的值。
- 对每个测试用例，该函数恰被调用一次。

上述函数可以调用以下函数，来指导 Amaru 进行交易：

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

- M：Amaru 支付给卖家的硬币数量。
- 该函数返回一对元素。第一个元素是数组 L，包含按顺序排列的已购买的纪念品种类。第二个元素是整数 R，表示交易后卖家返还给 Amaru 的硬币数量。
- 要求 $P[0] > M \geq P[N-1]$。条件 $P[0] > M$ 确保 Amaru 不购买第 0 种纪念品，而条件 $M \geq P[N-1]$ 确保他至少购买一件纪念品。如果这些条件未被满足，你将收到 `Output isn't correct: Invalid argument`。注意，与 $P[0]$ 不同，$P[N-1]$ 的值未在输入中提供 。
- 对每个测试用例，该函数最多被调用 5000 次。

评测程序的行为是**非自适应的**。 这意味着在调用 `buy_souvenirs` 前，价格序列 P 是固定的。

约束条件

- $2 \leq N \leq 100$
- 对每个满足 $0 \leq i < N$ 的 i，有 $1 \leq P[i] \leq 10^{15}$ 。
- 对每个满足 $0 \leq i < N-1$ 的 i，有 $P[i] > P[i+1]$ 。

子任务

子任务	分数	额外的约束条件
1	4	$N = 2$
2	3	对每个满足 $0 \leq i < N$ 的 i，有 $P[i] = N - i$ 。
3	14	对每个满足 $0 \leq i < N-1$ 的 i，有 $P[i] \leq P[i+1] + 2$。
4	18	$N = 3$
5	28	对每个满足 $0 \leq i < N-2$ 的 i，有 $P[i+1] + P[i+2] \leq P[i]$。 对每个满足 $0 \leq i < N-1$ 的 i，有 $P[i] \leq 2 \cdot P[i+1]$。
6	33	没有额外的约束条件。

例子

考虑以下函数调用。

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

共有 $N = 3$ 种纪念品，且 $P[0] = 4$。观察到只有三种可能的价格序列 P：$[4, 3, 2]$，$[4, 3, 1]$ 和 $[4, 2, 1]$。

假设 buy_souvenirs 调用 `transaction(2)`，且函数返回 $([2], 1)$，表示 Amaru 购买了一件第 2 种纪念品，而且卖家返还了 1 枚硬币。通过观察，我们可以推断出 $P = [4, 3, 1]$，因为：

- 对于 $P = [4, 3, 2]$，`transaction(2)` 会返回 $([2], 0)$。
- 对于 $P = [4, 2, 1]$，`transaction(2)` 会返回 $([1], 0)$。

然后 buy_souvenirs 可以调用 `transaction(3)` 返回 $([1], 0)$，表示 Amaru 购买了一件第 1 种纪念品，而卖家返还了 0 枚硬币。到目前为止，Amaru 购买了一件第 1 种纪念品和一件第 2 种纪念品。

最后，buy_souvenirs 可以调用 `transaction(1)` 返回 $([2], 0)$，表示 Amaru 购买了一件第 2 种纪念品。注意，这里也可以调用 `transaction(2)`。至此，Amaru 共购买了一件第 1 种纪念品和两件第 2 种纪念品，符合要求。

评测程序示例

输入格式：

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

输出格式：

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

这里，对每个满足 $0 \leq i < N$ 的 i，购买的第 i 种纪念品的数量为 $Q[i]$。