Hack! (hack)

Codeforces 比賽經過左個零鐘，你留意到同你一間房嘅另一個參賽者用咗 `unordered_set` 去解題。係時候 hack 佢喇！

你知道個 `unordered_set` 係一個 hash table，有 n 個 bucket，由 0 去到 $n-1$ 號。但係衰在你唔知個 n 係幾多，想搵返佢出嚟。

當你隊個整數 x 入個 hash table 嘅時候，佢會去咗 $(x \bmod n)$ 嗰個 bucket 度。如果隊入去之前，嗰個 bucket 已經有 b 樣嘢嘅話，就會引起 b 次 hash collision。

你可以畀 k 個唔同嘅整數 $x[0], x[1], \ldots, x[k-1]$ 落去個 interactor 度，就可以知道喺整一個裝住呢啲數嘅 `unordered_set` 嗰陣，總共發生咗幾多次 hash collision。不過呢，一次過畀 k 個整數畀個 interactor 做一個 query，就會用咗 k 嘅 cost。

例如，如果 $n = 5$，隊 $x = [2, 15, 7, 27, 8, 30]$ 呢啲數入個 interactor，總共會有 4 次 collision：

動作	新 collision	buckets
一開始	$-$	$[], [], [], [], []$
隊 $x[0] = 2$	0	$[], [], [2], [], []$
隊 $x[1] = 15$	0	$[15], [], [2], [], []$
隊 $x[2] = 7$	1	$[15], [], [2, 7], [], []$
隊 $x[3] = 27$	2	$[15], [], [2, 7, 27], [], []$
隊 $x[4] = 8$	0	$[15], [], [2, 7, 27], [8], []$
隊 $x[5] = 30$	1	$[15, 30], [], [2, 7, 27], [8], []$

留意番，個 interactor 係順住次序咁隊啲嘢入一個一開始係空嘅 `unordered_set` 度，嚟整嗰個 hash table，而且每一次 query 都會整一個新嘅空 `unordered_set`。換句話講，所有 query 都係獨立嘅。

你要做嘅野就係要喺總共用咗最多 $1\ 000\ 000$ cost 嘅情況下，搵返個 n 出嚟，即係有幾多個 bucket。

點樣 Implement

你要寫以下呢個 procedure：

```
int hack()
```

- 呢個 procedure 要 return 一個整數 – 就係收埋咗嗰個 n。
- 每一個 test case，個 grader 可能會 call 呢個 function 多過一次。每一次 call 都當做一個全新嘅情況去處理。

喺呢個 procedure 入面，你可以 call 以下呢個 procedure：

```
long long collisions(std::vector<long long> x)
```

- x: 一個裝住唔同數嘅 array，入面每一個 $x[i]$ 都係 $1 \leq x[i] \leq 10^{18}$。
- 呢個 function 會 return 番將 x 入面啲嘢隊入一個 `unordered_set` 會產生嘅 collision 數量。
- 呢個 procedure 可以 call 好多次。喺一次 call `hack()` 入面，所有 call 嘅 x 嘅長度加埋，唔可以多過 $1\ 000\ 000$。

注意：**因為個 `hack()` procedure 會 call 多過一次，參賽者要留意番上一個 call 剩低嘅 data 對下一個 call 嘅影響，尤其係 global variable 嘅狀態。**

嗰個 $1\ 000\ 000$ 嘅 cost 上限係 apply 去每一個 test case。一般嚟講，如果 call `hack()` t 次，你總共可以用最多 $t \times 1\ 000\ 000$ 嘅 cost，而每一次 call `hack()` 用嘅 cost 唔可以多過 $1\ 000\ 000$。

個 interactor 唔係 adaptive 嘅，即係話 n 嘅數值係喺開始互動之前已經定死咗。

例子

假設，有 2 個 test case。個 grader 會 call：

```
hack()
```

假設，喺個 function 入面，你 call：

Call	Returned value
`collisions([2, 15, 7, 27, 8, 30])`	4
`collisions([1, 2, 3])`	0
`collisions([10, 20, 30, 40, 50])`	10

之後，如果你搵到個 n 係 5，咁個 `hack()` procedure 就要 return 5。

跟住個 grader 會再 call 多一次：

```
hack()
```

假設，喺個 function 入面，你 call：

Call	Returned value
`collisions([1, 3])`	1
`collisions([2, 4])`	1

唯一一個 n 滿足到啲 query 嘅就係 2。所以個 `hack()` procedure 應該要 return 2。

限制

- $1 \leq t \leq 10$，個 t 就係有幾多個 test case。
- $2 \leq n \leq 10^9$
- 每一次 call `collisions()`，入面每一個 $x[i]$ 都係 $1 \leq x[i] \leq 10^{18}$。

Subtask

1. (8 分) $n \leq 500\ 000$
2. (17 分) $n \leq 1\ 000\ 000$
3. (75 分) 冇額外限制。

最後一個 subtask 有部分分數。假設 q 係喺呢個 subtask 入面，所有 test case 嘅所有 `hack()` call 嘅最高總 cost。你喺呢個 subtask 嘅分數係根據以下呢個表去計：

條件	分數
$1\ 000\ 000 < q$	0
$110\ 000 < q \leq 1\ 000\ 000$	$75 \cdot \log_{50}\left(\frac{10^6}{x-90000}\right)$
$q \leq 110\ 000$	75

如果喺是但一個 test case 入面，call `collisions()` 唔符合「點樣 Implement」嗰度講嘅限制，或者個 `hack()` return 番個數唔啱，咁你喺嗰個 subtask 嘅分數就會係 0。

Sample Grader

個 sample grader 會讀以下呢個 format 嘅 input：

- 第一行：t

跟住會有 t 行，每一行都係一個 n 嘅數值：

- 第一行：n

每一個 test case，假設 m 係 `hack()` return 番個數，而 c 就係所有 query 嘅總 cost。個 sample grader 會 print 你個答案，format 如下：

- 第一行：$m\ c$