Hack! (hack)

Llevas una hora compitiendo en un Codeforces, te das cuenta que un concursante resolvió un problema usando un `unordered_set`. Es hora de hackearle!

Tu sabes que un unordered set usa una tabla hash con n cubetas (buckets), las cuales están enumeradas del 0 al $n-1$. Desafortunadamente, tú no conoces el valor de n y quieres encontrarlo.

Cuando insertas un entero x en la tabla hash, se inserta a la $(x \bmod n)$ -iésima cubeta. Si habian b elementos en esta cubeta antes de la inserción, el nuevo elemento causará b colisiones de hash.

Dándole k enteros distintos $x[0], x[1], \ldots, x[k-1]$ al interactor, puedes encontrar el número total de colisiones de hash que ocurrieron al crear el set que contiene a dichos números. Sin embargo, darle al interactor k enteros en una query tiene un costo de k.

Por ejemplo, si $n = 5$, darle al interactor $x = [2, 15, 7, 27, 8, 30]$ causa 4 colisiones en total:

Operación	Nuevas colisiones	Cubetas
inicialmente	$-$	$[], [], [], [], []$
inserta $x[0] = 2$	0	$[], [], [2], [], []$
inserta $x[1] = 15$	0	$[15], [], [2], [], []$
inserta $x[2] = 7$	1	$[15], [], [2, 7], [], []$
inserta $x[3] = 27$	2	$[15], [], [2, 7, 27], [], []$
inserta $x[4] = 8$	0	$[15], [], [2, 7, 27], [8], []$
inserta $x[5] = 30$	1	$[15, 30], [], [2, 7, 27], [8], []$

Notar que el interactor crea la tabla hash insertando los elementos en orden en un conjunto inicialmente vacío, y se creará un nuevo conjunto vacío para cada query. En otras palabras, todas las queries son independientes.

Tu tarea es encontrar el número n de cubetas utilizando un costo total de como máximo $1\,000\,000$.

Detalles de implementación

Deberás implementar la siguiente función:

```
int hack()
```

- Esta función deberá retornar un entero – el valor oculto de n.
- Para cada caso de prueba, se podrá llamar a esta función más de una vez. Cada llamada deberá ser procesada como un nuevo escenario.

Dentro de la función anterior, podrás llamar a la siguiente función:

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

- x: un arreglo de números distintos, donde $1 \leq x[i] \leq 10^{18}$ para cada i.
- Esta función retorna el número de colisiones creadas al insertar los elementos de x al unordered set.
- Esta función puede ser llamada múltiples veces. La suma de la longitud de x para todas las llamadas dentro de una llamada de `hack()` no deberá exceder $1\ 000\ 000$.

Nota: **Como la función `hack()` se llamará más de una vez, los competidores deberán prestar atención al impacto de las variables globales de la anterior llamada.**

El límite de $1\ 000\ 000$ de costo aplica para cada caso de prueba. En general, si hay t llamadas a `hack()`, deberás usar un costo de no más de $t \times 1\ 000\ 000$, con cada llamada individual a `hack()` usando un costo de no más de $1\ 000\ 000$.

El interactor no es adaptativo, es decir, los valores de n son fijos al desde el inicio de la interacción.

Example

Supón, que hay 2 casos de prueba. El evaluador hará la siguiente llamada:

```
hack()
```

Digamos que dentro de la función tu haces las siguientes llamadas:

Llamada	Valor retornado
`collisions([2, 15, 7, 27, 8, 30])`	4
`collisions([1, 2, 3])`	0
`collisions([10, 20, 30, 40, 50])`	10

Después de eso, si encuentras que el valor de n es 5, la función `hack()` deberá retornar 5.

Después el evaluador hará otra llamada:

```
hack()
```

Digamos que dentro de la función tu haces las siguientes llamadas:

Llamada	Valor retornado
`collisions([1, 3])`	1
`collisions([2, 4])`	1

La única n que cumple con las queries es 2. Entonces, la función `hack()` deberá retornar 2.

Límites

- $1 \leq t \leq 10$, donde t es el número de casos de prueba.
- $2 \leq n \leq 10^9$
- $1 \leq x[i] \leq 10^{18}$ para cada llamada a `collisions()`.

Subtareas

1. (8 puntos) $n \leq 500\ 000$
2. (17 puntos) $n \leq 1\ 000\ 000$
3. (75 puntos) Sin restricciones adicionales.

En la última subtarea puedes obtener un puntaje parcial. Sea q el máximo costo entre todas las llamadas a `hack()` en una subtarea. Tu puntaje para esta subtarea se calcula de acuerdo a la siguiente tabla:

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

Si, en cualquiera de los casos de prueba, las llamadas a la función `collisions()` no cumplen con los límites descritos en los detalles de implementación, o el número retornado por la función `hack()` no es correcto, tu puntaje para dicha subtarea será 0.

Evaluador de ejemplo

El evaluador de ejemplo leerá los datos de la siguiente forma:

- línea 1: t

Después, t líneas que describen un valor de n:

- línea 1: n

Para cada caso de prueba, sea m el valor de retorno de `hack()`, y c sea el costo total de todas las queries. El evaluador de ejemplo te imprimirá la respuesta en el siguiente formato:

- línea 1: $m\ c$