Hack! (hack)

หลังจากการแข่งขัน Codeforces ผ่านไปหนึ่งชั่วโมง คุณสังเกตว่าผู้เข้าแข่งขันคนอื่นในห้องเดียวกับคุณแก้ปัญหาโดยใช้ `unordered_set` ได้เวลามา hack แล้ว!

คุณทราบว่า `unordered_set` ใช้ตารางแฮช (hash table) ที่มีช่อง n ช่อง โดยมีหมายเลขจาก 0 ถึง $n-1$ อย่างไรก็ตาม โชคไม่ดีที่คุณไม่ทราบค่า n และต้องการจะหาว่าค่าดังกล่าวเป็นเท่าใด

เมื่อคุณใส่จำนวนเต็ม x ลงในตารางแฮช จำนวนเต็มนั้นจะถูกใส่ลงในช่องที่ $(x \bmod n)$ ถ้ามีข้อมูลจำนวน b ตัวอยู่ที่ช่องดังกล่าว การใส่ข้อมูลนี้จะทำให้เกิดการชนของแฮช b ครั้ง

ถ้าคุณส่งจำนวนเต็ม k ที่แตกต่างกัน $x[0], x[1], \ldots, x[k-1]$ ให้กับส่วนติดต่อกับตารางแฮช (the interactor) คุณจะทราบจำนวนครั้งของการชนของแฮชที่เกิดขึ้นทั้งหมดในการสร้าง `unordered_set` ที่ประกอบไปด้วยจำนวนเต็มเหล่านี้ อย่างไรก็ตาม การส่งจำนวนเต็ม k ตัวให้กับส่วนติดต่อกับตารางแฮช ก็จะมีค่าใช้จ่าย k เช่นเดียวกัน

พิจารณาตัวอย่างต่อไปนี้ สมมติให้ $n = 5$ การส่งรายการ $x = [2, 15, 7, 27, 8, 30]$ ให้กับส่วนติดต่อกับตารางแฮชจะทำให้เกิดการชนของแฮชจำนวนรวม 4 ครั้ง ดังนี้:

การทำงาน	จำนวนการชนของแฮชที่เกิดขึ้น	ช่อง
initially	$-$	$[], [], [], [], []$
insert $x[0] = 2$	0	$[], [], [2], [], []$
insert $x[1] = 15$	0	$[15], [], [2], [], []$
insert $x[2] = 7$	1	$[15], [], [2, 7], [], []$
insert $x[3] = 27$	2	$[15], [], [2, 7, 27], [], []$
insert $x[4] = 8$	0	$[15], [], [2, 7, 27], [8], []$
insert $x[5] = 30$	1	$[15, 30], [], [2, 7, 27], [8], []$

สังเกตว่า ส่วนติดต่อกับตารางแฮชจะสร้างตารางแฮชโดยการใส่จำนวนในรายการไปตามลำดับ ในตาราง `unordered_set` ที่ว่างเปล่า จะมีการสร้าง `unordered_set` ขึ้นมาใหม่ทุกครั้ง สำหรับแต่ละคำถาม นั่นคือ ทุก ๆ การถามจะเป็นอิสระต่อกัน

งานของคุณคือหาจำนวนช่อง n ของตารางแฮช โดยใช้ค่าใช้จ่ายไม่เกิน $1\ 000\ 000$

รายละเอียดการเขียนโปรแกรม

คุณจะต้องเขียนฟังก์ชันต่อไปนี้:

```
int hack()
```

- ฟังก์ชันจะต้องคืนจำนวนเต็มหนึ่งจำนวนแทนค่าของ n
- ในแต่ละชุดข้อมูลทดสอบ เกรดเดอร์อาจจะเรียกฟังก์ชันนี้มากกว่าหนึ่งครั้ง สำหรับการเรียกแต่ละครั้ง ให้พิจารณาว่าเป็นสถานการณ์ที่แตกต่างกัน

ในฟังก์ชันดังกล่าว คุณสามารถเรียกฟังก์ชันด้านล่างได้:

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

- x: อาร์เรย์ของจำนวนเต็มที่แตกต่างกันโดยที่ $1 \leq x[i] \leq 10^{18}$ สำหรับแต่ละ i
- ฟังก์ชันนี้จะคืนค่าจำนวนการชนของแฮชทั้งหมดที่เกิดขึ้นการใส่ข้อมูลใน x ลงใน `unordered_set`
- สามารถเรียกฟังก์ชันนี้ได้หลายครั้ง ผลรวมของความยาวของ x รวมในการเรียกจากฟังก์ชัน `hack()` ที่ถูกเรียกหนึ่งครั้งจะต้องไม่เกิน $1\ 000\ 000$.

หมายเหตุ: **เนื่องจากฟังก์ชัน `hack()` จะถูกเรียกมากกว่าหนึ่งครั้ง ผู้เข้าแข่งขันจะต้องดูแลและจัดการกับผลกระทบจากข้อมูลที่ตกค้างจากการเรียกในครั้งก่อนต่อการเรียกในครั้งปัจจุบัน โดยเฉพาะข้อมูลในตัวแปรโกลบอล (global variables)**

ขีดจำกัดค่าใช้จ่าย $1\ 000\ 000$ นั้นจะพิจารณาสำหรับแต่ละกรณีทดสอบ โดยทั่วไป ถ้ามีการ `hack()` จำนวน t ครั้ง คุณจะสามารถใช้ค่าใช้จ่ายได้รวมไม่เกิน $t \times 1\ 000\ 000$ โดยที่ในแต่ละการเรียก `hack()` จะใช้ค่าใช้จ่ายได้ไม่เกิน $1\ 000\ 000$

ส่วนติดต่อกับตารางแฮชจะไม่ทำงานแบบปรับเปลี่ยนได้ (ไม่ adaptive) นั่นคือ ค่าของ n จะถูกกำหนดไว้ก่อนจะเริ่มการทำงานเสมอ

ตัวอย่าง

สมมติว่ามีชุดทดสอบ 2 ชุด เกรดเดอร์จะเรียก

```
hack()
```

สมมติว่าภายในการเรียกฟังก์ชัน `hack` ดังกล่าว คุณเรียกฟังก์ชันดังนี้

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

หลังจากนั้น ถ้าคุณพบว่าค่าของ n คือ 5 ฟังก์ชัน `hack()` จะต้องคืนค่า 5

จากนั้น เกรดเดอร์จะเรียกฟังก์ชันอีกครั้งหนึ่ง:

```
hack()
```

สมมติว่าภายในการเรียกฟังก์ชัน `hack` ดังกล่าว คุณเรียกฟังก์ชันดังนี้

การเรียก	ค่าที่คืนกลับมา
`collisions([1, 3])`	1
`collisions([2, 4])`	1

ค่าของ n ที่สอดคล้องกับการเรียกคือ 2 ดังนั้นฟังก์ชัน `hack()` จะต้องคืนค่า 2

เงื่อนไข

- $1 \leq t \leq 10$, โดยที่ t คือจำนวนของกรณีทดสอบ
- $2 \leq n \leq 10^9$
- $1 \leq x[i] \leq 10^{18}$ สำหรับแต่ละการเรียกใช้ `collisions()`

ปัญหาย่อย

1. (8 points) $n \leq 500\ 000$
2. (17 points) $n \leq 1\ 000\ 000$
3. (75 points) ไม่มีเงื่อนไขเพิ่มเติมอื่น ๆ

ในปัญหาย่อยสุดท้าย คุณสามารถได้คะแนนบางส่วนได้ ให้ q แทนค่าใช้จ่ายมากที่สุดตลอดการเรียกใช้ฟังก์ชัน `hack()` ของทุก ๆ กรณีทดสอบในปัญหาย่อยนี้ คะแนนของคุณสำหรับปัญหาย่อยนี้จะถูกคำนวณตามตารางด้านล่างนี้:

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

ถ้าในกรณีทดสอบใด ๆ การเรียกฟังก์ชัน `collisions()` ไม่เป็นไปตามเงื่อนไขตามที่ระบุในส่วน "รายละเอียดการเขียนโปรแกรม" หรือค่าที่คืนจาก `hack()` ไม่ถูกต้อง คะแนนของปัญหาย่อยนั้นจะเป็น 0

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างจะอ่านข้อมูลนำเข้าในรูปแบบดังนี้:

- บรรทัด 1: t

จากนั้นจะมีข้อมูลอีก t บรรทัด แต่ละบรรทัดระบุค่า n:

- บรรทัด 1: n

สำหรับแต่ละกรณีทดสอบ ให้ m แทนค่าที่คืนจาก `hack()` และ c แทนค่าใช้จ่ายทั้งหมด เกรดเดอร์ตัวอย่างจะพิมพ์คำตอบของคุณในรูปแบบดังนี้:

- line 1: m c