האק! (hack)

שעה לתוך תחרות Codeforces, שמתם לב שמתחרה אחר בחדר שלכם פתר בעיה באמצעות `unordered_set`. זמן לעשות האק!

אתם יודעים ש-`unordered_set` משתמש בטבלת גיבוב (hash table) בעלת n תאים, הממוספרים מ-0 עד $n-1$. למרבה הצער, אתם לא יודעים את הערך של n ומעוניינים לשחזר אותו.

כשאתם מכניסים מספר שלם x לטבלת הגיבוב, הוא מוכנס לתא ה- $(x \bmod n)$. אם ישנם b איברים בתא זה לפני ההכנסה, זה יגרום ל-b התנגשויות גיבוב להתרחש.

באמצעות מתן k מספרים שלמים שונים $x[0], x[1], \ldots, x[k-1]$ לאינטראקטור, אתם יכולים למצוא את המספר הכולל של התנגשויות גיבוב שהתרחשו בעת יצירת `unordered_set` המכיל את המספרים. אבל, הזנת האינטרקטור ב-k מספרים שלמים בשאילתה אחת תגרור עלות של k.

לדוגמה, אם $n = 5$, הזנת האינטרקטור ב-$x = [2, 15, 7, 27, 8, 30]$ תגרום ל-4 התנגשויות בסך הכל:

פעולה	התנגשויות חדשות	תאים
מצב התחלתי	$-$	$[], [], [], [], []$
הכנס $x[0] = 2$	0	$[], [], [2], [], []$
הכנס $x[1] = 15$	0	$[15], [], [2], [], []$
הכנס $x[2] = 7$	1	$[15], [], [2, 7], [], []$
הכנס $x[3] = 27$	2	$[15], [], [2, 7, 27], [], []$
הכנס $x[4] = 8$	0	$[15], [], [2, 7, 27], [8], []$
הכנס $x[5] = 30$	1	$[15, 30], [], [2, 7, 27], [8], []$

שימו לב שהאינטרקטור יוצר את טבלת הגיבוב על ידי הכנסת האיברים לפי הסדר ל-`unordered_set` שריק בהתחלה, וש-`unordered_set` ריק חדש יווצר עבור כל שאילתה. במילים אחרות, כל השאילתות בלתי תלויות.

משימתכם היא למצוא את מספר התאים n תוך שימוש בעלות כוללת של לכל היותר $1\ 000\ 000$.

פרטי מימוש

עליכם לממש את הפונקציה הבאה:

```
int hack()
```

- על פונקציה זו להחזיר מספר שלם – הערך הנסתר של n.
- בכל טסט, הגריידר יכול לקרוא לפונקציה זו יותר מפעם אחת. יש לעבד כל קריאה כתרחיש חדש נפרד.

מתוך פונקציה זו, אתם יכולים לקרוא לפונקציה הבאה:

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

- x: מערך של מספרים שונים, כאשר $1 \leq x[i] \leq 10^{18}$ לכל i.
- פונקציה זו מחזירה את מספר ההתנגשויות הנוצרות כתוצאה מהכנסת איברי x ל-`unordered_set`.
- פונקציה זו יכולה להיקרא מספר פעמים. אסור שסכום האורכים של x בכל הקריאות מתוך קריאה אחת ל-`hack()` יחרוג מעבר ל-$1\ 000\ 000$.

שימו לב: **משום שהפונקציה `hack()` תיקרא יותר מפעם אחת, על המתחרים לשים לב להשפעת הנתונים שנשארו מהקריאה הקודמת על הקריאה הנוכחית, במיוחד למצב המאוכסן במשתנים גלובליים.**

מגבלת העלות של $1\ 000\ 000$ חלה על כל טסט. באופן כללי, אם יש t קריאות ל-`hack()`, מותר לכם להשתמש בעלות כוללת של לא יותר מ-$t \times 1\ 000\ 000$, כאשר כל אחת מהקריאות ל-`hack()` בנפרד משתמשת בעלות של לא יותר מ-$1\ 000\ 000$.

האינטראקטור אינו אדפטיבי, כלומר ערכי n נקבעים לפני תחילת האינטרקציה.

דוגמה

הניחו שיש 2 קריאות. הגריידר יבצע את הקריאה הבאה:

```
hack()
```

נניח שמתוך הפונקציה אתם מבצעים את הקריאות הבאות:

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

לאחר מכן, אם אתם מגלים שהערך של n הוא 5, הפונקציה `hack()` צריכה להחזיר 5.

אחר כך הגריידר יבצע קריאה נוספת:

```
hack()
```

נניח שמתוך הפונקציה אתם מבצעים את הקריאות הבאות:

קריאה	ערך החזרה
`collisions([1, 3])`	1
`collisions([2, 4])`	1

ערך ה-n היחיד שמתאים לשאילתות הוא 2. לכן, על הפונקציה `hack()` להחזיר 2.

אילוצים

- $1 \leq t \leq 10$, כאשר t הוא מספר הקריאות.
- $2 \leq n \leq 10^9$
- $1 \leq x[i] \leq 10^{18}$ עבור כל קריאה ל-`collisions()`.

תת משימות

1. (8 נקודות) $n \leq 500\ 000$
2. (17 נקודות) $n \leq 1\ 000\ 000$
3. (75 נקודות) ללא אילוצים נוספים.

בתת המשימה האחרונה, אתם יכולים לקבל ניקוד חלקי. תהי q העלות הכוללת המירבית מבין כל הקריאות ל-`hack()`, על פני כל הטסטים של תת המשימה. הניקוד שלכם בתת משימה זו מחושב לפי הטבלה הבאה:

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

אם, בטסט כלשהו, הקריאות לפונקציה `collisions()` לא תואמות לאילוצים המתוארים בפרק "פרטי מימוש", או שהמספר המוחזר על ידי `hack()` שגוי, הניקוד של הפתרון שלכם לתת המשימה הזו יהיה 0.

גריידר לדוגמה

הגריידר לדוגמה קורא את הקלט בפורמט הבא:

- שורה 1: t

לאחר מכן, מופיעות t שורות, שכל אחת מהן מכילה ערך של n:

- שורה 1: n

עבור כל טסט, יהי m ערך ההחזרה של `hack()`, ו-c העלות הכוללת של כל השאילתות. הגריידר לדוגמה מדפיס את התשובה שלכם בפורמט הבא:

- שורה 1: $m\ c$