แบ่งกลุ่ม

เตมีอร์และอูลุคเบค ผู้ช่วยของเขา กำลังเตรียมมายากลสำหรับรายการ *Uzbekistan Got Talent* โดยเตมีอร์จะแสดงการแก้**ปัญหาแบ่งกลุ่ม**ดังนี้: แบ่งชุดจำนวนเต็มบวกที่ให้มา ออกเป็นกลุ่มที่ไม่ว่างเปล่าจำนวน K กลุ่ม โดยให้ผลรวมของสมาชิกในแต่ละกลุ่มมีค่าเท่ากัน กล่าวคือ ทุกจำนวนที่ให้มาจะต้องอยู่ใน 1 กลุ่มพอดีจากทั้งหมด K กลุ่ม และผลรวมของแต่ละกลุ่มจะต้องเท่ากัน ตัวอย่างเช่น ถ้าให้ชุดจำนวน $[2, 1, 6, 4, 5]$ มา และให้ $K = 3$ การแบ่งกลุ่มที่ถูกต้องคือ $[2, 4]$, $[1, 5]$ และ $[6]$ ซึ่งผลรวมของสมาชิกในแต่ละกลุ่มจะเท่ากับ 6 พอดี

วิธีแสดงมายากลมีดังนี้

- อูลุคเบคและเตมีอร์เข้าไปอยู่ในคนละห้อง ซึ่งไม่สามารถติดต่อสื่อสารกันได้
- กรรมการมอบอาร์เรย์ A ของจำนวนเต็มบวก N จำนวน $A[0], A[1], \ldots, A[N-1]$ ให้อูลุคเบค โดยแต่ละจำนวนมีค่าอยู่ในช่วง 1 ถึง M (รวมหัวท้าย) และกรรมการได้มอบค่า K ให้อูลุคเบคด้วย
- อูลุคเบคเลือกจำนวนเต็มไม่เกิน $K-1$ จำนวน (ไม่จำเป็นต้องแตกต่างกัน) เพื่อ**ใส่เพิ่ม**เข้าไปในอาร์เรย์ A ซึ่งจำนวนที่เพิ่มเข้ามาใหม่ก็จะต้องมีค่าอยู่ในช่วง 1 ถึง M (รวมหัวท้าย) เช่นกัน
- กรรมการใส่จำนวนเต็มที่อูลุคเบคเลือกมาต่อท้ายอาร์เรย์เดิม จากนั้นอาร์เรย์ที่ขยายแล้วนี้จะถูกนำไป**จัดเรียงให้เป็นลำดับไม่ลด** แล้วจึงมอบให้เตมีอร์ พร้อมกับค่า K
- จากนั้น เตมีอร์จะต้องแก้ปัญหาแบ่งกลุ่ม สำหรับอาร์เรย์ที่ขยายและจัดเรียงแล้วนี้

งานของคุณคือการวางกลยุทธ์ให้เตมีอร์และอูลุคเบค เราสามารถพิสูจน์ได้ว่ามันมีกลยุทธ์ที่ทำให้ทั้งสองคนสามารถแก้ปัญหาแบ่งกลุ่มนี้ได้เสมอโดยไม่ขึ้นอยู่กับค่าของอาร์เรย์ A ที่กรรมการมอบให้

รายละเอียดการเขียนโปรแกรม

คุณต้องเขียนฟังก์ชัน 2 ฟังก์ชันต่อไปนี้

ฟังก์ชันที่คุณต้องเขียนสำหรับ**อูลุคเบค**คือ

```
std::vector<int> add_numbers(std::vector<int> A, int K, int M)
```

- A: อาร์เรย์ความยาว N ที่มอบให้อูลุคเบคในตอนแรก
- K: จำนวนกลุ่มที่ต้องแบ่ง สังเกตว่าอูลุคเบคสามารถใส่จำนวนเต็มเพิ่มได้ไม่เกิน $K-1$ จำนวน
- M: ค่าสูงสุดที่อนุญาต สำหรับจำนวนเต็มที่ให้มา และจำนวนเต็มที่จะใส่เพิ่ม
- ฟังก์ชันนี้จะถูกเรียกใช้เพียงครั้งเดียวต่อข้อมูลทดสอบ

ฟังก์ชันนี้จะต้องคืนค่าอาร์เรย์ C ที่ประกอบด้วยจำนวนเต็มที่อูลุคเบคต้องการใส่เพิ่ม ให้ S แทนความยาวของอาร์เรย์ C

- S ต้องมีค่าไม่เกิน $K-1$
- แต่ละจำนวนในอาร์เรย์ C ต้องมีค่าอยู่ในช่วง 1 ถึง M (รวมหัวท้าย)

ฟังก์ชันที่คุณต้องเขียนสำหรับ**เตมือร์**คือ

```
std::vector<int> find_partition(std::vector<int> B, int K)
```

- B: อาร์เรย์ความยาว $N + S$ ที่ประกอบด้วยจำนวนเต็มที่อยู่ใน A ตั้งแต่แรก และจำนวนเต็มที่อูลุคเบคใส่เพิ่มเข้ามา อาร์เรย์ B ได้ถูกจัดเรียงให้เป็นลำดับไม่ลดมาแล้ว
- K: จำนวนกลุ่มที่ต้องแบ่ง
- ฟังก์ชันนี้จะถูกเรียกใช้เพียงครั้งเดียวต่อข้อมูลทดสอบ

ฟังก์ชันนี้จะต้องคืนค่าอาร์เรย์ P ที่แสดงถึงการแบ่งกลุ่ม B ออกเป็น K กลุ่ม

- ความยาวของ P ต้องเป็น $N + S$
- สำหรับแต่ละ j ที่ $0 \leq j < N + S$ นั้น $P[j]$ จะบ่งบอกถึงกลุ่มที่ $B[j]$ เป็นสมาชิก
- กลุ่มจะมีหมายเลขตั้งแต่ 0 ถึง $K - 1$ นั่นหมายความว่า $0 \leq P[j] < K$ สำหรับทุก $0 \leq j < N + S$
- สำหรับแต่ละ i ตั้งแต่ 0 ถึง $K - 1$ จะต้องมี j ($0 \leq j < N + S$) อย่างน้อย 1 ตัวที่ $P[j] = i$.
- ผลรวมของสมาชิกในแต่ละกลุ่ม จะต้องเท่ากันหมดทั้ง K กลุ่ม

ในการตรวจให้คะแนนจริง โปรแกรมที่เรียกใช้ฟังก์ชันดังกล่าวจะถูกรัน **2 ครั้งพอดี**

- ในการรันครั้งแรก
  - `add_numbers` จะถูกเรียกใช้ 1 ครั้งพอดี
  - ระบบเกรดเดอร์จะประมวลผลอาร์เรย์ที่ถูกคืนค่ามา เพื่อนำไปคำนวณอาร์เรย์ B
- ในการรันครั้งที่สอง
  - `find_partition` จะถูกเรียกใช้ 1 ครั้งพอดี

เงื่อนไข

- $3 \leq N \leq 100\,000$
- $2 \leq K \leq 100\,000$
- $K \leq N$
- $1 \leq M \leq 10^9$
- $1 \leq A[i] \leq M$ สำหรับแต่ละ i ที่ $0 \leq i < N$.

ปัญหาย่อย

ปัญหาย่อย	คะแนน	เงื่อนไขเพิ่มเติม
1	5	$N = 3$
2	4	$M = 1$
3	7	$M \leq 2$
4	10	$N \leq 10$
5	7	$K = 2$
6	12	$K \leq 3$
7	17	$K \leq 10$
8	20	$K \leq 100$
9	18	ไม่มีเงื่อนไขเพิ่มเติม

ตัวอย่าง

พิจารณาการเรียกต่อไปนี้

```
add_numbers([8, 2, 9, 6, 1, 5, 5], 3, 9)
```

ในตัวอย่างนี้ $A = [8, 2, 9, 6, 1, 5, 5]$ เราต้องการแบ่งจำนวนออกเป็น $K = 3$ กลุ่ม และอูลุคเบคสามารถใส่เพิ่มจำนวนที่มีค่าอยู่ในช่วง 1 ถึง $M = 9$

ฟังก์ชันสามารถคืนค่า $[5, 4]$ หมายความว่าอูลุคเบคใส่จำนวนเพิ่มมา $S = 2$ จำนวน ได้แก่ 5 และ 4 นี่เป็นการคืนค่าที่ถูกต้อง เนื่องจากทั้งสองจำนวนมีค่าอยู่ในช่วง 1 ถึง $M = 9$.

อาร์เรย์ที่ขยายแล้วได้ถูกนำไปจัดเรียงและส่งมอบให้กับเตมีอร์ในการเรียก

```
find_partition([1, 2, 4, 5, 5, 5, 6, 8, 9], 3)
```

เราสามารถแบ่งจำนวนในอาร์เรย์ดังกล่าวออกเป็น 3 กลุ่ม ได้แก่ $[1, 5, 9]$, $[2, 5, 8]$ และ $[4, 5, 6]$ สังเกตว่าผลรวมของสมาชิกในแต่ละกลุ่มมีค่าเท่ากับ 15
ในการตอบวิธีแบ่งกลุ่มดังกล่าว ฟังก์ชันจะต้องคืนค่าอาร์เรย์ $[0, 1, 2, 0, 1, 2, 2, 1, 0]$

เกรดเดอร์ตัวอย่าง

รูปแบบข้อมูลนำเข้า:

```
N K M
A[0] A[1] ... A[N-1]
```

รูปแบบข้อมูลส่งออก:

หลังจากเรียกใช้ฟังก์ชัน `add_numbers` เสร็จแล้ว เกรดเดอร์จะ

- คำนวณอาร์เรย์ B ที่เรียงลำดับแล้ว
- พิมพ์อาร์เรย์ C และ B ในรูปแบบต่อไปนี้ **ตามด้วยบรรทัดว่าง**

```
S
C[0] C[1] ... C[S-1]
B[0] B[1] ... B[N+S-1]
```

หลังจากเรียกใช้ฟังก์ชัน `find_partition` เสร็จแล้ว เกรดเดอร์จะแสดงผล

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

ในที่นี้ L คือความยาวของอาร์เรย์ P ที่คืนค่ามาจากฟังก์ชัน `find_partition`