בעיית חלוקה

תומר ועוזרו, סושרד, מכינים קסם לתוכנית אוזבקיסטן גוט טאלנט. החלק המרכזי בקסם הוא פתירת **בעיית החלוקה** הבאה על ידי תומר: יש לחלק אוסף נתון של מספרים שלמים חיוביים ל-K קבוצות לא ריקות כך שהסכום של האיברים בכל קבוצה זהה בין כל הקבוצות. במילים אחרות, כל מספר שלם מהאוסף חייב להיות משויך לבדיוק אחת מ-K הקבוצות, והסכום בתוך כל קבוצה חייב להיות זהה. לדוגמה, אם האוסף הנתון הוא $[2, 1, 6, 4, 5]$ ו-$K = 3$, חלוקה חוקית יכולה להיות $[2, 4]$, $[1, 5]$, ו-$[6]$. במקרה הזה, סכום האיברים בכל קבוצה הוא 6 בדיוק.

הקסם מבוצע בצורה הבאה:

- תומר וסושרד נכנסים לדוכנים שונים ולא יכולים לתקשר אחד עם השני.
- שופט מביא לסושרד מערך A של N מספרים שלמים חיוביים $A[0], A[1], \ldots, A[N-1]$, שבו כל מספר בין 1 ל-$M$ כולל. השופט גם מביא לסושרד את הערך של K.
- סושרד בוחר לכל היותר $K - 1$ מספרים שלמים (לא בהכרח שונים) **להוספה כאיברים חדשים** למערך A. כל מספר שמוסיפים חייב גם הוא להיות בין 1 ל-M כולל.
- השופט מוסיף את המספרים שנבחרו על ידי סושרד למערך המקורי. המערך המורחב לאחר מכן **ממוין בסדר לא יורד** ומועבר לתומר, יחד עם הערך K.
- תומר חייב לפתור את בעיית החלוקה עבור המערך המורחב והממוין הזה.

המשימה שלכם היא לתכנן ולממש אסטרגיה עבור תומר וסושרד. ניתן להוכיח, שבאילוצים הנתונים, קיימת אסטרטגיה עבור תומר וסושרד כדי לפתור בהצלחה את בעית החלוקה, ללא תלות במערך A המסופק על ידי השופטים.

פרטי מימוש

עליכם לממש את הפונקציות הבאות:

הפונקציה שעליכם לממש עבור **סושרד** היא:

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

- A: מערך באורך N המייצג את המערך המקורי שניתן לסושרד.
- K: מספר הקבוצות הרצוי; שימו לב שסושרד יכול להוסיף עד $K - 1$ מספרים שלמים למערך.
- M: הערך המירבי המותר עבור כל מספר מקורי וחדש במערך.
- הפונקציה הזאת נקראת בדיוק פעם אחת עבור כל מקרה בדיקה.

הפונקציה צריכה להחזיר מערך C שכולל את המספרים שסושרד רוצה להוסיף למערך המקורי. נסמן ב-S את האורך של המערך C.

- S חייב להיות לכל היותר $K - 1$.

- כל איבר ב-C חייב להיות בין 1 ל-M כולל.

הפונקציה שעליכם לממש עבור **תומר** היא:

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

- B: מערך באורך $N+S$ הכולל גם את המספרים המקוריים מהמערך A וגם את המספרים שסושרד הוסיף. האיברים של המערך B ממוינים בסדר לא יורד.
- K: מספר הקבוצות הרצוי.
- הפונקציה הזאת נקראת פעם אחת בדיוק בכל מקרה בדיקה.

הפונקציה צריכה להחזיר מערך P שמייצג את החלוקה של B ל-K קבוצות זרות.

- האורך של P צריך להיות $N+S$.
- עבור כל j המקיים $0 \le j < N+S$, $P[j]$ מייצג את הקבוצה אשר $B[j]$ שייך אליה.
- הקבוצות צריכות להיות ממוספרות מ-0 עד $K-1$, כלומר $0 \le P[j] < K$ חייב להתקיים עבור כל $0 \le j < N+S$.
- עבור כל i בין 0 ל-$K-1$ כולל, חייב להיות איבר אחד לפחות j ($0 \le j < N+S$) כך ש-$P[j] = i$.
- סכום האיברים ששויכו לכל אחת מ-K הקבוצות צריך להיות זהה.

בזמן הבדיקה האמיתי, תוכנה שתיקרא לפונקציות לעיל תרוץ **בדיוק פעמיים**.

- במהלך הריצה הראשונה של התוכנה:
  - `add_numbers` תיקרא בדיוק פעם אחת.
  - המערך שיוחזר יעובד על ידי מערכת הניקוד כדי לחשב את המערך B.
- במהלך הריצה השניה של התוכנה:
  - `find_partition` תיקרא בדיוק פעם אחת.

אילוצים

- $3 \le N \le 100\,000$
- $2 \le K \le 100\,000$
- $K \le N$
- $1 \le M \le 10^9$
- $1 \le A[i] \le M$ לכל i המקיים $0 \le i < N$.

תתי משימות

תת משימה	ניקוד	אילוצים נוספים
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	ללא אילוצים נוספים.

דוגמה

נתבונן בקריאה הבאה:

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

בדוגמה זו $A = [8, 2, 9, 6, 1, 5, 5]$. אנחנו רוצים לחלק את המספרים ל-$K = 3$ קבוצות. סושרד יכול להוסיף מספרים בין 1 ל-$M = 9$.

הפונקציה יכולה להחזיר $[5, 4]$, כלומר שסושרד החליט להוסיף $S = 2$ איברים חדשים: 5 ו-4. הם חוקיים כי שניהם בין 1 לבין $M = 9$.

המערך המורחב כעת ממויין ומועבר לתומר עם הקריאה הבאה:

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

אנו יכולים לחלק את האיברים ממערך זה לשלושת הקבוצות הבאות: $[1, 5, 9]$, $[2, 5, 8]$, ו-$[4, 5, 6]$. שימו לב שהסכום של כל קבוצה שווה ל-15. כדי לדווח על החלוקה הזו, על הפונקציה להחזיר את המערך $[0, 1, 2, 0, 1, 2, 2, 1, 0]$.

גריידר לדוגמה

פורמט קלט:

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

פורמט פלט:

אחרי שהקריאה ל-`add_numbers` מסתיימת, הגריידר לדוגמה:

- יחשב את המערך הממוין B.
- ידפיס את המערכים C ו-B בפורמט הבא, **ולאחריהם שורה ריקה**:

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

אחרי שהקריאה ל-`find_partition` מסתיימת, הגריידר לדוגמה מדפיס כפלט:

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

כאן, L הוא האורך של המערך P שמוחזר על ידי `find_partition`.