Lotteria di quadri (`quadri`)

Facendo pulizia in soffitta Fabio ha ritrovato diversi quadri, alcuni del tutto privi di valori e altri invece potenzialmente interessanti (tra cui diverse copie di alcuni dipinti di Leonardo da Vinci). Per racimolare qualche soldo, ha intenzione di venderli: ha pertanto portato a valutare gli N quadri da un esperto che ha attribuito a ciascuno un valore V_i approssimativo.

Per far sì che tutti i quadri di minor valore non restino invenduti, continunando a ingombrare la soffitta, Fabio è intenzionato a organizzare una lotteria con una regola particolare: le opere saranno disposte in fila e l'acquirente, dopo aver pagato un "biglietto di ingresso", potrà scegliere a propria discrezione un "blocco" consecutivo di B opere (non di più né di meno).



Figura 1: Alcuni dei quadri esposti (foto di Muhammad Raufan Yusup).

Fabio non vuole che la somma dei valori delle opere che un acquirente può portarsi a casa sia maggiore di un certo massimale M, altrimenti ci starebbe perdendo troppo. D'altro canto, pur rispettando questo principio, vorrebbe rendere B (il numero di quadri che ci si porta a casa comprando il biglietto d'ingresso) più alto possibile. Aiutalo a capire qual è il valore massimo di B per rendere il più appetibile possibile la lotteria!

Implementazione

Dovrai sottoporre un unico file con estensione `.cpp` o `.c`.

> ☞ Tra gli allegati a questo task troverai un template (`quadri.cpp` e `quadri.c`) con un esempio di implementazione.

Dovrai implementare la seguente funzione:

■ Funzione `quadri`

C/C++	`int quadri(int N, long long M, int V[]);`

- L'intero N rappresenta il numero dei quadri.
- L'intero M rappresenta il massimale da non superare.
- L'array `V`, indicizzato da 0 a $N-1$, contiene alla posizione i il valore dell'i-esimo quadro nella fila.
- La funzione dovrà restituire il valore massimo per B come descritto nel testo.

Il grader chiamerà la funzione `quadri` e ne stamperà il valore restituito sul file di output.

Allegata a questo problema è presente una versione semplificata del grader usato durante la correzione, che potete usare per testare le vostre soluzioni in locale. Il grader di esempio legge i dati da `stdin`, chiama la funzione che dovete implementare e scrive su `stdout`, secondo il seguente formato.

Il file di input è composto da due righe, contententi:

- Riga 1: gli interi N e M, separati da uno spazio.
- Riga 2: N interi `V[`i`]` per $i = 0, \ldots, N-1$.

Il file di output è composto da un'unica riga, contenente:

- Riga 1: il valore restituito dalla funzione `quadri`.

Assunzioni

- $1 \leq N \leq 200\,000$.
- $1 \leq M \leq 10^{12}$.
- $1 \leq V_i \leq 10^6$ per ogni $i = 0 \ldots N-1$.
- Nella scelta del blocco di B quadri, l'acquirente **non** può scegliere un blocco agli estremi della fila in modo da prenderne meno di B.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1** [**0 punti**]: Casi d'esempio.
- **Subtask 2** [**15 punti**]: $M < V_i$ per ogni $i = 0, \ldots, N-1$ oppure $M > V_0 + V_1 + \ldots + V_{N-1}$.
- **Subtask 3** [**20 punti**]: $N \leq 500$.
- **Subtask 4** [**25 punti**]: $N \leq 5\,000$.
- **Subtask 5** [**40 punti**]: Nessuna limitazione specifica.

Esempi di input/output

input	output
4 8 1 2 3 4	2
5 1 3 1 3 2 10	0

Spiegazione

Nel **primo caso di esempio** è possibile impostare $B = 2$: l'acquirente potrebbe scegliere i quadri di valore $1 + 2 = 3$, $2 + 3 = 5$ oppure $3 + 4 = 7$ senza superare mai il massimale $M = 8$. Non sarebbe stato possibile scegliere $B = 3$ perché in quel caso l'acquirente avrebbe potuto il scegliere il blocco di valore $2 + 3 + 4 = 9$, superando il massimale.

Nel **secondo caso di esempio** anche scegliere $B = 1$ consentirebbe all'acquirente di superare il massimale qualunque scelta egli faccia (ad eccezione del secondo quadro). La risposta corretta è quindi $B = 0$.