

cAPS IOCK (capslock)

Gabriele ha scritto un messaggio per Giorgio, ma arrivato alla fine si è accorto con orrore che tutto il testo ha le minuscole e le maiuscole invertite per colpa del caps lock attivo.

Piuttosto che riscrivere tutto daccapo, Gabriele decide di chiederti di creare un programma che, preso il testo del messaggio, converta le maiuscole in minuscole e viceversa.

Aiuta Gabriele ad aggiustare il messaggio!

Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene l'unico intero N , il numero di caratteri del testo. La seconda riga contiene il testo del messaggio.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente il testo corretto.

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📎 Tra gli allegati a questo task troverai un template (`capslock.c`, `capslock.cpp`, `capslock.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>void aggiusta(int N, char S[]);</code>
Pascal	<code>procedure aggiusta(N: longint; var S: array of char);</code>

In cui:

- L'intero N rappresenta il numero di caratteri del testo.
- L'array S , indicizzato da 0 a $N - 1$, contiene i caratteri di cui il testo è composto. Questi possono essere:
 - una lettera (maiuscola o minuscola) dell'alfabeto;
 - uno spazio;
 - un segno di punteggiatura tra questi: `.,:;!?.`
- La funzione dovrà trasformare la stringa S nel testo corretto. Al termine dell'esecuzione della funzione verrà stampato sul file di output il nuovo contenuto dell'array S .

Assunzioni

- $1 \leq N \leq 10\,000$.



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 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $1 \leq N \leq 100$.
- **Subtask 3 [40 punti]:** $1 \leq N \leq 1000$.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
41 eHI, TUTTO BENE? tI VA UNA PIZZA STASERA?	Ehi, tutto bene? Ti va una pizza stasera?
input.txt	output.txt
23 nA nA nA nA nA bATmAN!!	Na Na Na Na Na BatMan!!

Biglietti a Milano (biglietti)

Giorgio, che studia a Torino, ha deciso di far visita a Gabriele, che studia a Milano. Giorgio sa che durante il periodo che passerà con Gabriele avrà bisogno di fare N viaggi sui mezzi pubblici, e per questo sta indagando sui prezzi dei biglietti. Ha scoperto che a Milano è possibile comprare un biglietto valido per una singola corsa per A centesimi, oppure un carnet da M viaggi, al costo di B centesimi.

Conoscendo N , M , A e B , quanti centesimi al minimo deve spendere Giorgio per poter fare N corse sui mezzi?

Dati di input

Il file `input.txt` è composto da un'unica riga contenente gli interi N, M, A, B .

Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📎 Tra gli allegati a questo task troverai un template (`biglietti.c`, `biglietti.cpp`, `biglietti.pas`) con un esempio di implementazione.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int compra(int N, int M, int A, int B);</code>
Pascal	<code>function compra(N, M, A, B: longint): longint;</code>

In cui:

- L'intero N rappresenta il numero di corse che Giorgio deve fare.
- L'intero M rappresenta il numero di corse che sono comprese in un carnet.
- L'intero A rappresenta il costo in centesimi di una corsa singola.
- L'intero B rappresenta il costo in centesimi di un intero carnet.
- La funzione dovrà restituire il minimo numero di centesimi che è necessario spendere, che verrà stampato sul file di output.

Assunzioni

- $1 \leq N, M, A, B \leq 10\,000$.
- È possibile che Giorgio compri un numero di corse maggiore di N , se conveniente.

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 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $1 \leq N \leq 100$.
- **Subtask 3 [40 punti]:** $1 \leq N \leq 1000$.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
4 10 150 1380	600
input.txt	output.txt
11 10 150 1380	1530
input.txt	output.txt
10 10 150 1700	1500
input.txt	output.txt
11 10 150 100	200

Spiegazione

Nel **primo caso di esempio** conviene comprare 4 biglietti singoli.

Nel **secondo caso di esempio** conviene comprare un carnet e un biglietto.

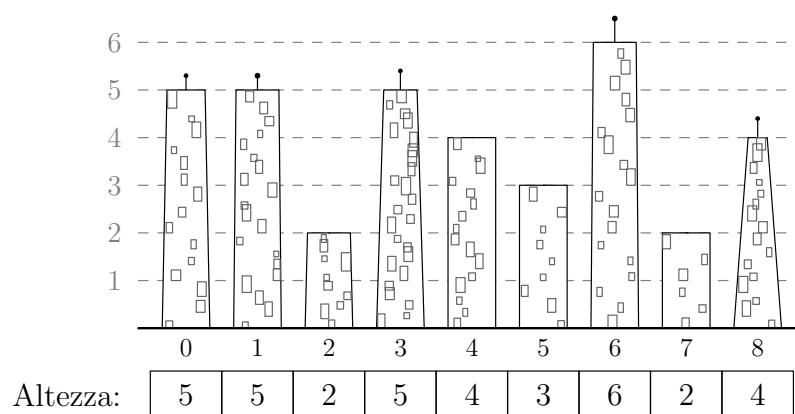
Nel **terzo caso di esempio** conviene comprare 10 biglietti singoli.

Nel **quarto caso di esempio** conviene comprare due carnet.

Fila di grattacieli (grattacieli)

Giorgio è in visita a Toronto, città famosa per il suo skyline di N grattacieli di varia altezza tutti disposti lungo un'unica fila. Giorgio è appassionato di grattacieli, e quindi intende salire su uno di essi e approfittare dell'altezza per ammirare il panorama (e cioè, gli altri grattacieli). Giorgio sa che dalla cima di un grattacielo ammirerà appieno tutti i grattacieli (compreso quello su cui si trova) che non sono coperti da un altro grattacielo. Un grattacielo viene coperto da un altro (più vicino di questo al grattacielo su cui Giorgio si trova) se questo altro grattacielo è sia alto almeno quanto il grattacielo che copre e sia alto almeno quanto il grattacielo su cui Giorgio si trova.

Ad esempio, consideriamo questo skyline:



In questo scenario, se Giorgio salisse sul grattacielo 3, di altezza 5, sarebbe in grado di osservare solo i grattacieli da 1 a 6 compresi. Infatti il grattacielo 0 è coperto dal grattacielo 1, alto 5; i grattacieli 7 e 8 sono coperti alla vista dal grattacielo 6.

Aiuta Giorgio a scegliere su quale grattacielo salire, calcolando qual è il massimo numero di grattacieli che Giorgio può ammirare se sceglie nel modo ottimo il grattacielo su cui salire.

Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene l'unico intero N . La seconda riga contiene N interi separati da uno spazio, le altezze H_i dei grattacieli.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📁 Tra gli allegati a questo task troverai un template (`grattacieli.c`, `grattacieli.cpp`, `grattacieli.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:



C/C++	<code>int osserva(int N, int H[]);</code>
Pascal	<code>function osserva(N: longint; var H: array of longint): longint;</code>

In cui:

- L'intero N rappresenta il numero di grattacieli presenti nella skyline.
- L'array H , indicizzato da 0 a $N - 1$, contiene le altezze H_i dei grattacieli nell'ordine in cui sono disposti lungo la skyline.
- La funzione dovrà restituire il massimo numero di grattacieli che è possibile ammirare appieno, che verrà stampato sul file di output.

Assunzioni

- $1 \leq N \leq 100\,000$.
- $1 \leq H_i \leq 1\,000\,000$ per ogni $i = 0 \dots N - 1$.

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 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $N \leq 10$.
- **Subtask 3 [40 punti]:** $N, H_i \leq 1000$.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
9 5 5 2 5 4 3 6 2 4	9
input.txt	output.txt
6 9 4 2 9 9 7	5

Spiegazione

Nel **primo caso di esempio**, corrispondente all'esempio del testo, è possibile ammirare appieno tutti i grattacieli salendo sul grattacielo 6, di altezza 6.

Nel **secondo caso di esempio**, è possibile ammirare al massimo 5 grattacieli, salendo sul grattacielo 3, di altezza 9.

Rifornimenti ai distributori (distributori)

Gabriele ha appena preso la patente, e decide di invitare tutti i suoi amici a fare una gita. Dato che il viaggio è lungo ben K chilometri, sa che forse dovrà fermarsi a fare il pieno di benzina: a tal proposito Gabriele ha segnato a che distanza dalla partenza ci sono gli N distributori che si trovano lungo il tragitto. Sapendo che la sua macchina fa al massimo M chilometri con un pieno, e che alla partenza ha già il serbatoio pieno, aiuta Gabriele a pianificare i rifornimenti di modo da fare benzina il minor numero possibile di volte.

Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene i tre interi N , M , K separati da uno spazio. La seconda riga contiene N interi separati da uno spazio, le distanze D_i dei distributori in ordine crescente.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📖 Tra gli allegati a questo task troverai un template (`distributori.c`, `distributori.cpp`, `distributori.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int rifornisci(int N, int M, int K, int D[]);</code>
Pascal	<code>function rifornisci(N, M, K: longint; var D: array of longint): longint;</code>

In cui:

- L'intero N rappresenta il numero di distributori.
- L'intero M rappresenta il massimo numero di chilometri con un pieno che la macchina è in grado di fare.
- L'intero K rappresenta la lunghezza totale del viaggio in chilometri.
- L'array D , indicizzato da 0 a $N - 1$, contiene le distanze dei distributori dalla partenza in ordine crescente.
- La funzione dovrà restituire il minor numero di rifornimenti che è necessario fare, che verrà stampato sul file di output.

Assunzioni

- $1 \leq N \leq 100\,000$.
- $1 \leq M \leq K \leq 1\,000\,000$.
- $1 \leq D_i < D_{i+1} < K$ per ogni $i = 0 \dots N - 2$.
- È sempre possibile raggiungere la destinazione: la distanza tra due distributori successivi non supera mai M , il numero di chilometri che la macchina è in grado di percorrere con un pieno.

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 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $N \leq 10$.
- **Subtask 3 [40 punti]:** $N, K \leq 1000$.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
5 50 100 29 35 50 77 83	1
input.txt	output.txt
10 30 100 1 31 33 38 62 69 93 97 98 99	6

Spiegazione

Nel **primo caso di esempio**, è sufficiente fermarsi al distributore al chilometro 50.

Nel **secondo caso di esempio**, è necessario fermarsi ai distributori ai chilometri 1, 31, 33 o 38, 62, 69, e infine uno tra quelli ai chilometri 93, 97, 98, 99.

Interrogazioni equilibrate (interrogazioni)

Giulia, la professoressa di storia dell'arte di Gabriele, ha deciso di interrogare K persone oggi. Rivolgerà ad ogni interrogato una domanda, presa da una lista (segreta!) di N quesiti di varia difficoltà. La professoressa sa bene che le domande rivolte agli studenti dovranno essere tutte di difficoltà comparabile, per evitare il malcontento della classe. La scontentezza della classe infatti è pari alla differenza tra la difficoltà della domanda più difficile e quella più facile tra quelle chieste.

Se Giulia sceglie K domande dalla lista nel modo ottimo, qual è la minima scontentezza possibile della classe?

Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene i due interi N e K separati da uno spazio. La seconda riga contiene N interi separati da uno spazio, le difficoltà D_i delle domande della lista.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📎 Tra gli allegati a questo task troverai un template (`interrogazioni.c`, `interrogazioni.cpp`, `interrogazioni.pas`) con un esempio di implementazione.

Dovrai implementare la seguente funzione:

C/C++	<code>int interroga(int N, int K, int D[]);</code>
Pascal	<code>function interroga(N, K: longint; var D: array of longint): longint;</code>

In cui:

- L'intero N rappresenta il numero totale di domande nella lista.
- L'intero K rappresenta il numero di domande da selezionare.
- L'array D , indicizzato da 0 a $N - 1$, contiene le difficoltà delle domande della lista.
- La funzione dovrà restituire la minima scontentezza possibile della classe, che verrà stampata sul file di output.

Assunzioni

- $1 \leq K \leq N \leq 10\,000$.
- $1 \leq D_i \leq 100\,000$ per ogni $i = 0 \dots N - 1$.

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 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $1 \leq N \leq 10$.
- **Subtask 3 [10 punti]:** $1 \leq N \leq 1000$, $K = 2$.
- **Subtask 4 [30 punti]:** $1 \leq N \leq 1000$.
- **Subtask 5 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
5 2 4 10 7 12 9	1

input.txt	output.txt
6 3 12 34 54 59 29 44	15

Spiegazione

Nel **primo caso di esempio** conviene selezionare le domande di difficoltà 9 e 10.

Nel **secondo caso di esempio** conviene selezionare le domande di difficoltà 34, 29 e 44 oppure le domande di difficoltà 54, 59 e 44.

Tagli al personale (licenziamenti)

La *SteamPower S.P.A.*, azienda leader mondiale nel campo delle macchine a vapore portatili, sta attraversando un brutto periodo di crisi e per tenerla in piedi nell'attesa che l'economia riparta la dirigenza ha stabilito un regime di forti tagli al personale.

Per stabilire chi deve essere licenziato, il reparto personale ha distribuito un test con cui ha stabilito accuratamente di ognuno degli N dipendenti il suo personale valore di bravura B_i . Inoltre si baserà anche sullo studio dell'organigramma (cioè il grafico ad albero in cui tutti i dipendenti dell'azienda sono organizzati secondo le relazioni di dipendenza), e conosce quindi il capo diretto C_i di ogni dipendente i . Per convenzione, il presidente dell'azienda è segnato al numero 0 e ha $C_i = -1$ per indicare l'assenza di un capo diretto.

Le direttive presidenziali hanno stabilito che per cominciare verranno licenziati tutti i dipendenti *inadequati*, e cioè quelli che hanno valore di bravura B_i strettamente maggiore di quello di uno dei loro capi (diretti o indiretti). Aiuta il reparto personale a determinare quante persone dovranno essere licenziate!

Dati di input

Il file `input.txt` è composto da $N + 1$ righe. La prima riga contiene l'unico intero N . Le successive N righe contengono ciascuna due interi separati da uno spazio, i valori B_i e C_i del dipendente i -esimo.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📎 Tra gli allegati a questo task troverai un template (`licenziamenti.c`, `licenziamenti.cpp`, `licenziamenti.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int licenzia(int N, int B[], int C[]);</code>
Pascal	<code>function licenzia(N: longint; var B, C: array of longint): longint;</code>

In cui:

- L'intero N rappresenta il numero totale di dipendenti dell'azienda.
- L'array B , indicizzato da 0 a $N - 1$, contiene i valori di bravura B_i dei dipendenti.
- L'array C , indicizzato da 0 a $N - 1$, contiene gli indici dei capi diretti C_i (sempre da 0 a $N - 1$) dei dipendenti corrispondenti. Il valore C_0 è garantito essere pari a -1 e indica che il presidente è sempre il dipendente numero 0.
- La funzione dovrà restituire il numero di dipendenti inadeguati, che verrà stampato sul file di output.

Assunzioni

- $1 \leq N \leq 100\,000$.
- $0 \leq B_i \leq 1\,000\,000$ per ogni $i = 0 \dots N - 1$.
- $0 \leq C_i \leq N - 1$ per ogni $i = 1 \dots N - 1$, mentre $C_0 = -1$.
- È garantito che l'organigramma rappresenti effettivamente un albero, e cioè che risalendo di capo in capo a partire da ogni dipendente i si arrivi sempre al presidente 0.

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.

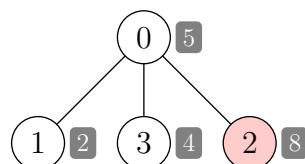
- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $N \leq 10$.
- **Subtask 3 [30 punti]:** $N \leq 100$.
- **Subtask 4 [10 punti]:** L'organigramma forma una linea retta, e quindi $C_i = i - 1$ per ogni i .
- **Subtask 5 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
<pre>4 5 -1 2 0 8 0 4 0</pre>	<pre>1</pre>
input.txt	output.txt
<pre>7 4 -1 2 0 1 4 2 4 6 0 3 1 5 4</pre>	<pre>3</pre>

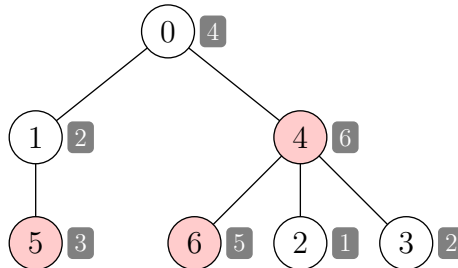
Spiegazione

Il **primo caso di esempio** corrisponde a questo organigramma, in cui i numeri bianchi nei riquadri scuri rappresentano il valore di bravura dei dipendenti:



Il dipendente 2 ha un valore di bravura superiore a quello del suo capo (il presidente dell'azienda), e per questo deve essere licenziato.

Il **secondo caso di esempio** corrisponde invece a questo organigramma:



I dipendenti 4 e 5 hanno un valore di bravura maggiore dei loro rispettivi capi, e quindi vanno licenziati. Il dipendente 6 ha un valore di bravura minore di quello del suo capo, ma maggiore di quello del capo del suo capo, e quindi va licenziato.

Numeri di Figonacci (figonacci)

Dopo l'ultimo seminario di teoria dei numeri, Giorgio è rimasto affascinato dallo studio della sequenza dei numeri di Fibonacci. Pertanto, per non essere da meno, introduce una nuova sequenza di numeri secondo lui ancora più interessante: i numeri di *Figonacci*. Come per i loro quasi-omonimi, l' $(n+1)$ -esimo numero di Figonacci G_{n+1} si calcola a partire dai precedenti (eccezion fatta per i primi due numeri di Figonacci, che sono valori fissati a $G_0 = -1$ e $G_1 = 0$). La regola che stabilisce il valore di G_{n+1} , tuttavia, è diversa da quella dei numeri di Fibonacci: G_{n+1} è pari alla somma di tutte le possibili differenze tra il numero di Figonacci immediatamente precedente e quelli ancora prima. In formule:

$$\begin{aligned} G_{n+1} &= \sum_{i=0}^{n-1} (G_n - G_i) \\ &= (G_n - G_{n-1}) + (G_n - G_{n-2}) + \dots + (G_n - G_2) + (G_n - G_1) + (G_n - G_0) \end{aligned}$$

I primi numeri che si ottengono da questa sequenza sono quindi $-1, 0, 1, 3, 9, \dots$ ed è facile vedere che crescono molto rapidamente. Ma questo non è un problema per Giorgio, che è interessato ad usarli per problemi di teoria dei numeri, e a cui quindi interessa soltanto il valore modulo M di questi numeri.

Aiuta la sua ricerca calcolando il valore dell' N -esimo numero di Figonacci G_N modulo M .

Dati di input

Il file `input.txt` è composto da un'unica riga contenente i due numeri interi N ed M .

Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

🔍 Tra gli allegati a questo task troverai un template (`figonacci.c`, `figonacci.cpp`, `figonacci.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int enumera(int N, int M);</code>
Pascal	<code>function enumera(N, M: longint): longint;</code>

In cui:

- L'intero N rappresenta l'indice del numero di Figonacci a cui Giorgio è interessato.
- L'intero M rappresenta il modulo con il quale va ridotto quel numero.
- La funzione dovrà restituire il valore di G_N modulo M , che verrà stampato sul file di output.



Assunzioni

- $2 \leq N \leq 1\,000\,000$.
- $2 \leq M \leq 40\,000$.

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 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $N \leq 10$.
- **Subtask 3 [40 punti]:** $N \leq 100$.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
3 10	3

input.txt	output.txt
4 3	0

input.txt	output.txt
5 9	6

Spiegazione

Nel **primo caso di esempio**, $G_3 = 3$ che modulo 10 resta 3.

Nel **secondo caso di esempio**, $G_4 = 9$ che modulo 3 fa 0.

Nel **terzo caso di esempio**, $G_5 = 33$ che modulo 9 fa 6.

Note

L'operazione di modulo si calcola con l'operatore `%` in C/C++ e `mod` in Pascal. Notare che in entrambi i casi può dare risultati non corretti se l'argomento è negativo (il che può succedere per diversi motivi). Un modo per risolvere questo problema è usare il seguente codice:

C/C++	$(GN \% M + M) \% M$
Pascal	$(GN \bmod M + M) \bmod M$

L'operazione di modulo, inoltre, ha le seguenti proprietà (molto utili per evitare *integer overflow* quando si vogliono calcolare numeri molto grandi):

- $(A + B) \bmod M = (A \bmod M + B \bmod M) \bmod M$
- $(A \cdot B) \bmod M = (A \bmod M \cdot B \bmod M) \bmod M$