Numeri di Figonacci (`figonacci`)

Limite di tempo: 1.0 secondi
Limite di memoria: 256 MiB

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:

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

I primi numeri che si ottengono da questa sequenza sono quindi $-1, 0, 1, 3, 9, \ldots$ 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++	`int enumera(int N, int M);`
Pascal	`function enumera(N, M: longint): longint;`

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 mod M + M) mod M`

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

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