

## Un problema di ricarica (ricarica)

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

Il cellulare di Giorgio ha la sua età e ormai si scarica molto velocemente. Per ogni minuto di utilizzo perde un punto di carica, proprio come per ogni minuto in cui è collegato alla corrente ne acquista uno. Se poi raggiunge zero punti di carica, va in *crash* e diventa necessario portarlo all'assistenza tecnica.

Giorgio deve ora uscire di casa per una serie di commissioni di una durata complessiva di  $M$  minuti, numerati da 1 a  $M$ . Per evitare di rimanere con il cellulare scarico, ha quindi stilato un elenco di tutti gli  $N$  intervalli di minuti  $(A_i, B_i)$  in cui sa che potrà collegarlo alla corrente e recuperare punti di carica. In tutti gli altri minuti, il cellulare perderà punti di carica.

Con quanti punti di carica Giorgio deve uscire di casa al minimo, per essere certo di non rimanere mai con il cellulare scarico in nessuno degli  $M$  minuti?

### Implementazione

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

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

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

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

In cui:

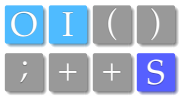
- L'intero  $N$  rappresenta il numero di intervalli in cui Giorgio collegherà il cellulare alla corrente.
- L'intero  $M$  rappresenta il numero di minuti totali delle commissioni.
- Gli array  $A$  e  $B$ , indicizzati da 0 a  $N - 1$ , contengono gli intervalli di minuti  $(A_i, B_i)$  in cui il cellulare sarà collegato alla corrente.
- La funzione dovrà restituire la minima quantità di punti di carica necessari affinché il cellulare non si scarichi durante gli  $M$  minuti, che verrà stampata sul file di output.

### Dati di input

Il file `input.txt` è composto da  $N + 1$  righe. La prima riga contiene i due interi  $N, M$ . La riga  $i + 1$  per  $i = 1 \dots N$  contiene i due interi  $A_i, B_i$  separati da uno spazio.

### Dati di output

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



## Assunzioni

- $1 \leq N \leq 100\,000$ .
- $1 \leq M \leq 2\,000\,000\,000$ .
- $1 \leq A_i \leq B_i \leq M$  per ogni  $i = 0 \dots N - 1$ .
- $B_i < A_{i+1}$  per ogni  $i = 0 \dots N - 2$ .

## 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]**:  $M \leq 10000$ .
- **Subtask 4 [30 punti]**: Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
5 15 1 1 5 6 8 8 9 9 12 14	3
3 100 7 16 28 29 30 60	15

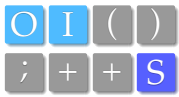
## Spiegazione

Nel **primo caso di esempio**, la carica del cellulare segue questo schema:

+	-	-	-	+	+	-	+	+	-	-	+	+	+	-
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

La carica minore viene quindi raggiunta alla fine del quarto minuto, in cui i punti residui saranno  $3 + 1 - 1 - 1 - 1 = 1$  e pertanto 3 è proprio la carica minima necessaria.

Nel **secondo caso di esempio**, se il cellulare parte con 15 punti di carica si ritrova con 9 all'inizio del settimo minuto, e poi con 19 alla fine del sedicesimo, con 8 all'inizio del ventottesimo, con 41 alla fine del sessantesimo e infine con 1 alla fine del centesimo.



## Impila la pila (combinazione)

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

William si è di recente appassionato a *Impila la pila*, un nuovo gioco per il suo telefonino. In questo gioco ci sono  $N$  interruttori in sequenza, numerati da 0 a  $N - 1$  da sinistra verso destra, e un personaggio che all'inizio è situato nell'angolo sinistro dello schermo (in corrispondenza dell'interruttore 0). Durante il gioco è possibile ordinare al personaggio di muoversi verso destra o verso sinistra (e queste operazioni richiedono un secondo) oppure di premere l'interruttore davanti al quale è situato (e questa operazione è istantanea).

A rendere il gioco interessante, sopra a ogni interruttore è posizionato un disco con una certa dimensione di base  $B_i$ . Ogni volta che viene premuto un interruttore, il disco corrispondente viene aggiunto a una pila che cresce quindi progressivamente. Lo scopo del gioco è di attivare tutti gli interruttori nell'ordine corretto, di modo che la pila che si viene a formare sia costituita da dischi con basi sempre strettamente decrescenti.

William ha appena finito un livello e non sa se affrontare anche quello successivo: tra pochi minuti deve presentarsi a una lezione, e non vuole assolutamente essere costretto a lasciarlo a metà. Aiutalo a calcolare di quanti secondi ha bisogno per completarlo, così che sia in grado di decidere se fa in tempo oppure no!

### Implementazione

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

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

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

C/C++	<code>long long premi(int N, int B[]);</code>
Pascal	<code>function premi(N: longint; var B: array of longint): int64;</code>

In cui:

- L'intero  $N$  rappresenta il numero di interruttori.
- L'array  $B$ , indicizzato da 0 a  $N - 1$ , contiene per ogni  $i$  la dimensione di base  $B_i$  del disco associato all'interruttore  $i$ -esimo.
- La funzione dovrà restituire il numero minimo di secondi necessario a completare il livello, che verrà stampato sul file di output.

### Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene l'unico intero  $N$ . La seconda riga contiene gli  $N$  interi  $B_i$  separati da uno spazio.

### Dati di output

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



## Assunzioni

- $1 \leq N \leq 100\,000$ .
- $1 \leq B_i \leq 100\,000$  per ogni  $i = 0 \dots N - 1$ .
- I valori  $B_i$  sono tutti differenti, ossia non esistono duplicati.

## 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]:**  $N \leq 1000$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
3 21 29 13	4
7 2 3 5 7 11 13 17	12

## Spiegazione

Nel **primo caso di esempio**, in un secondo si raggiunge l'interruttore con base 29, in un altro secondo l'interruttore con base 21 e in altri due secondi l'interruttore con base 13.

Nel **secondo caso di esempio**, in 6 secondi si raggiunge l'interruttore con base 17, e in altri 6 secondi si scandiscono tutti gli altri interruttori da destra verso sinistra.

## Truffa contabile (truffa)

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

La *SteamPower S.P.A.*, azienda leader mondiale nel campo delle macchine a vapore portatili, è ancora una volta in crisi nonostante le oculate manovre messe in atto nell'anno passato. Ora è il momento di stilare il bilancio di fine anno, che è di nuovo in passivo. Per non turbare gli azionisti, il *CEO* ha ricontattato il massimo esperto mondiale in campo di finanza creativa (il cui nome non possiamo rivelare).

Questa volta l'esperto ha elaborato un nuovo stratagemma: con un'audace manovra detta "*la sfangata*" una voce in uscita può diventare una voce in entrata. Questa manovra può essere ripetuta fino a che il bilancio non diventi in attivo, ma per minimizzare i rischi conviene effettuarla il *minor numero di volte*. Quante volte al minimo è necessario effettuare una sfangata affinché il bilancio diventi in attivo?

### Implementazione

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

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

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

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

In cui:

- L'intero  $N$  rappresenta il numero di voci del bilancio.
- L'array  $V$ , indicizzato da  $0$  a  $N - 1$ , contiene le voci  $V_i$  del bilancio (positive le entrate e negative le uscite).
- La funzione dovrà restituire il minor numero possibile di sfangate necessarie a rendere il bilancio in attivo, che verrà stampato sul file di output.

### Dati di input

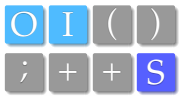
Il file `input.txt` è composto da due righe. La prima riga contiene l'unico intero  $N$ . La seconda riga contiene gli  $N$  interi  $V_i$  separati da uno spazio.

### Dati di output

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

### Assunzioni

- $1 \leq N \leq 100\,000$ .
- $-10\,000 \leq V_i \leq 10\,000$ ,  $V_i \neq 0$  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]:**  $N \leq 1000$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
5 -5000 -10000 3000 500 -4000	1
10 100 -1000 -5000 200 -2000 400 -1800 -400 -3000 50	2
2 -700 700	1

## Spiegazione

Nel **primo caso di esempio**, è sufficiente “sfangare” la voce  $-10\,000$  in  $10\,000$ .

Nel **secondo caso di esempio**, si possono ad esempio “sfangare” le voci  $-5000$  e  $-1800$ .

Nel **terzo caso di esempio**, si deve “sfangare” la voce  $-700$ .



## Zeri di coda (zeri)

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

Giorgio sta tenendo una lezione di Teoria dei grafi ai bambini della scuola elementare di Pinerolo, però non si è reso conto che a quell'età generalmente è difficile concentrarsi e prestare attenzione, soprattutto con un argomento complicato.

Quando alcuni dei bambini cominciano a chiacchierare, Giorgio si inquieta e gli assegna un difficile calcolo da svolgere per punizione. Il calcolo da svolgere è scritto su una lunga striscia di carta, sui cui c'è scritta una stringa che rappresenta una moltiplicazione. Ad esempio:

25x420x50x42

L'obiettivo è quello di *calcolare il numero di zeri in coda* al risultato della moltiplicazione. In questo caso, dato che la moltiplicazione ha come risultato 22050000, il numero di zeri in coda sarà pari a 4.

### Implementazione

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

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

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

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

In cui:

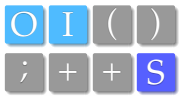
- L'intero  $N$  rappresenta la lunghezza della stringa.
- L'array  $S$ , indicizzato da  $0$  a  $N - 1$ , contiene la stringa con la moltiplicazione.
- La funzione dovrà restituire il numero di zeri in coda, che verrà stampato sul file di output.

### Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene l'unico intero  $N$ . La seconda riga contiene la stringa.

### Dati di output

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



## Assunzioni

- $1 \leq N \leq 100\,000$ .
- Tutte le sequenze di cifre separate dal carattere  $x$  rappresentano un numero compreso tra 1 e 1 000 000 estremi inclusi. Inoltre, non sono presenti zeri all'inizio delle varie sequenze (non sarà presente una sequenza come 0123).
- Nella striscia di carta saranno presenti 0 o più caratteri  $x$ .

## Assegnazione del punteggio

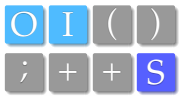
Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [40 punti]:** Il risultato della moltiplicazione si può memorizzare con un intero a 64 bit con segno.
- **Subtask 3 [30 punti]:** Tutte le varie sequenza della striscia *non* terminano con la cifra 5.
- **Subtask 4 [20 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
25x420x50x42	4
17264x72635x928x7x61537x8347x95475x8374	3





## Super Marco (scrigni)

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

William sta giocando al suo gioco preferito: Super Marco 64. Per superare un particolare livello all'interno del gioco, è necessario aprire una serie di scrigni nell'ordine corretto.

Il livello è fatto così: ci sono  $n$  scrigni numerati da 1 a  $n$ , ma William non sa quale scrigno corrisponde a quale numero. Se si cerca di aprire uno scrigno nell'ordine sbagliato, Super Marco riceve una scarica elettrica (e lo scrigno non si apre). Gli scrigni aperti fino a quel momento rimangono aperti.

Per esempio, supponiamo che ci siano 3 scrigni numerati così: 3, 1, 2. William, non conoscendo l'ordine, decide di aprire uno scrigno a caso. Per esempio, supponiamo che provi ad aprire quello più a sinistra. Lo scrigno scelto non è il numero 1, quindi non si aprirà e Super Marco prenderà la scossa. Ora William apre lo scrigno di mezzo, che è quello giusto. Prova di nuovo con quello più a sinistra (prendendo di nuovo la scossa), poi prova con quello più a destra (l'ultimo rimasto), e infine apre lo scrigno più a sinistra.

William ha quindi fatto prendere la scossa a Super Marco per ben 2 volte. La domanda però è la seguente: *in media*, quante volte si prende la scossa prima di trovare l'ordine corretto? Ad esempio, nel caso appena citato in cui  $n = 3$ , si può dimostrare facilmente che la scossa si prende in media 1.5 volte.

### Implementazione

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

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

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

C/C++	<code>double scosse(int N);</code>
Pascal	<code>function scosse(N: longint): double;</code>

In cui:

- L'intero  $N$  rappresenta il numero di scrigni.
- La funzione dovrà restituire il numero medio di scosse, che verrà stampato sul file di output.

### Dati di input

Il file `input.txt` è composto da una sola riga che contiene l'unico intero  $N$ .

### Dati di output

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



## Assunzioni

- La risposta verrà considerata corretta se l'errore assoluto o relativo non supererà  $10^{-6}$ .

## 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 [40 punti]:**  $1 \leq N \leq 8$ .
- **Subtask 3 [20 punti]:**  $1 \leq N \leq 12$ .
- **Subtask 4 [20 punti]:**  $1 \leq N \leq 100$ .
- **Subtask 5 [10 punti]:**  $1 \leq N \leq 2\,000\,000\,000$ .

## Esempi di input/output

input.txt	output.txt
3	1.5
4	3

## Shopping stress (negozi)

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

William odia fare shopping. C'è da dire però che non è il solo ad avere questo problema: una recente ricerca di mercato ha rilevato infatti che il 28% della popolazione cercherebbe qualsiasi scusa pur di non andare al centro commerciale!

Una startup italiana, venuta a sapere di questa ricerca, ha deciso di cogliere la palla al balzo: svilupperà un'app per rendere più sopportabile lo shopping alle persone pigre come William. Purtroppo però, nessuno dei programmatori iOS/Android che lavorano alla startup è riuscito a trovare un algoritmo efficiente per risolvere questo problema. Sapresti aiutarli?

Un centro commerciale è formato da una serie di  $N$  negozi in fila, numerati da 0 a  $N - 1$ . Ogni negozio è di una certa tipologia, identificata da un numero intero non negativo. L'app funziona così: William si trova in uno dei negozi (ad esempio, il negozio 7), si rende conto che deve comprare una felpa per l'inverno (supponiamo che le felpe si vendano nei negozi di tipo 13), William lancerà quindi l'app, la quale risponderà con un valore  $d$ : la distanza minima da percorrere per raggiungere un negozio di tipo 13 partendo dal negozio 7.


**Nota:** nel caso in cui il negozio 7 sia già un negozio di tipo 13, la distanza restituita sarà 0.

Il tuo algoritmo dovrà quindi rispondere a  $Q$  query, ognuna delle quali ti fornirà una coppia di valori  $(a, b)$ , rispettivamente: la posizione corrente ed il tipo di negozio desiderato. La risposta del tuo algoritmo sarà: la distanza tra la posizione  $a$  e la posizione del negozio di tipo  $b$  più vicino ad  $a$ .

**Nota:** le query sono indipendenti, l'effetto di una query non influisce sulle successive.

## Implementazione

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

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

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

C/C++	<code>void risolvi(int N, int Q, int t[], int a[], int b[], int d[]);</code>
Pascal	<code>procedure risolvi(N, Q: longint; var t, a, b, d: array of longint);</code>

In cui:

- L'intero  $N$  rappresenta il numero di negozi.
- L'intero  $Q$  rappresenta il numero di query a cui si deve rispondere.
- L'array  $t$ , indicizzato da 0 a  $N - 1$ , contiene il tipo di ciascun negozio.
- L'array  $a$ , indicizzato da 0 a  $Q - 1$ , contiene la "posizione attuale" da usare nella query.
- L'array  $b$ , indicizzato da 0 a  $Q - 1$ , contiene il "tipo di destinazione" da usare nella query.
- La funzione dovrà scrivere le risposte alle query nell'array  $d$ , indicizzato da 0 a  $Q - 1$ , che verrà stampato sul file di output.



## Dati di input

Il file `input.txt` è composto da  $Q + 2$  righe. La prima riga contiene due interi separati da spazio:  $N, Q$ . La seconda riga contiene gli  $N$  interi  $t_i$  separati da uno spazio. Seguono  $Q$  righe che contengono due interi ciascuna: i parametri  $a_i$  e  $b_i$  dell' $i$ -esima query.

## Dati di output

Il file `output.txt` è composto da  $Q$  righe. La  $i$ -esima riga contiene la risposta alla  $i$ -esima query.

## Assunzioni

- $1 \leq N, Q \leq 100\,000$ .
- $0 \leq t_i < 100\,000$ .
- $0 \leq a_i \leq N - 1$ .
- È sempre garantito che esiste almeno un negozio di tipo  $b_i$ .

## 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 [40 punti]:**  $N, Q \leq 5000$ .
- **Subtask 3 [30 punti]:**  $N \leq 5000$ .
- **Subtask 4 [20 punti]:** Nessuna limitazione specifica.

**Esempi di input/output**

input.txt	output.txt
6 3 0 1 2 1 2 0 2 0 3 0 1 2	2 2 1
10 6 23 5 29 3 3 5 2 7 11 13 0 5 3 5 5 5 9 5 0 3 9 3	1 2 0 4 3 5
2 3 10 20 0 10 0 20 1 10	0 1 1



## Robot camminatore (robot)

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

William ha finito di costruire un robot camminatore, che si muove su una griglia  $N \times M$  leggendo i caratteri scritti su di essa e interpretandoli come comandi. In particolare, reagisce in questo modo:

- X: sta fermo;
- +: effettua un passo (e cioè si muove di una casella) nella direzione in cui è orientato;
- 0: effettua un salto lungo due passi nella direzione in cui è orientato;
- L: ruota di  $90^\circ$  a sinistra e poi effettua un passo nella direzione in cui ora è orientato;
- R: ruota di  $90^\circ$  a destra e poi effettua un passo nella direzione in cui ora è orientato.

Nella griglia non sono presenti altri caratteri, né il robot reagisce ad altri comandi oltre a questi. Inoltre, la griglia su cui si muove è *toroidale*: vale a dire, ogni qualvolta il robot uscirebbe dal lato destro rientra da quello sinistro (e viceversa), e ogni qualvolta il robot uscirebbe dal lato superiore rientra da quello inferiore (e viceversa).

William ha ora intenzione di lasciare il robot nella casella  $(0, 0)$  orientato verso est, e vedere che percorso seguirà. Tuttavia, è preoccupato dall'eventualità che il percorso possa essere ciclico. Aiutalo verificando se il percorso seguito dal robot è ciclico, e se sì calcolando quanto è lunga la sua parte periodica (cioè che si ripete)!

## Implementazione

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

📁 Tra gli allegati a questo task troverai un template (`robot.c`, `robot.cpp`, `robot.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 M, matrix T);</code>
Pascal	<code>function osserva(N, M: longint; var T: matrix): longint;</code>

In cui:

- Gli interi  $N$ ,  $M$  rappresentano le due dimensioni della griglia.
- La matrice  $T$ , indicizzata da  $0$  a  $N - 1$  e da  $0$  a  $M - 1$ , contiene i caratteri presenti su ogni quadretto.
- La funzione dovrà restituire la lunghezza della parte periodica seguita dal robot (o il valore  $-1$  se il percorso non è periodico), che verrà stampata sul file di output.

## Dati di input

Il file `input.txt` è composto da  $N + 1$  righe. La prima riga contiene i due interi  $N$  e  $M$ . Le successive  $N$  righe contengono  $M$  interi ciascuna. In particolare l' $i$ -esima di queste righe contiene  $T[i][j]$  per  $j = 0 \dots M - 1$  separati da uno spazio.



## Dati di output

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

## Assunzioni

- $1 \leq N, M \leq 500$ .
- $T[i][j]$  è un carattere tra 'X+OLR', per ogni  $i = 0 \dots N - 1, j = 0 \dots M - 1$ .
- La lunghezza della parte periodica va calcolata in numero di passi (e quindi un salto conta 2).

## 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.

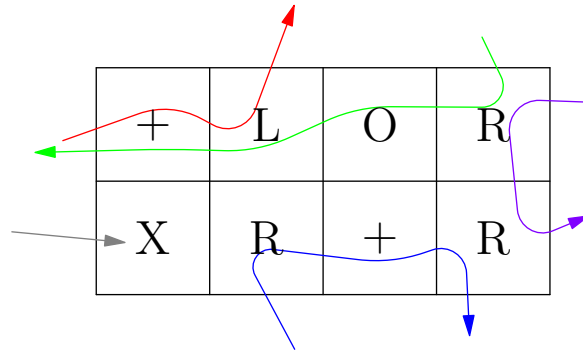
- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** Sono presenti solo i caratteri 'X+0'.
- **Subtask 3 [20 punti]:** Sono presenti solo i caratteri 'LR'.
- **Subtask 4 [10 punti]:** Sono presenti solo i caratteri '+LR'.
- **Subtask 5 [20 punti]:** Sono presenti solo i caratteri '+OLR'.
- **Subtask 6 [20 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

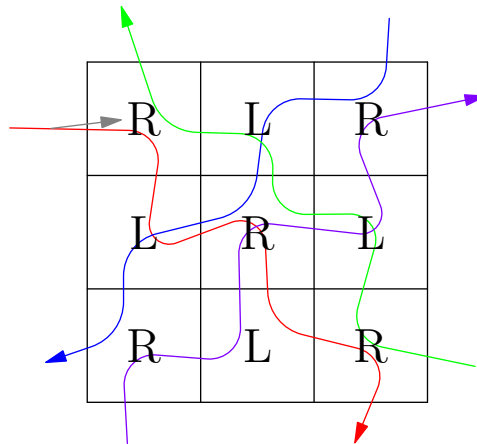
input.txt	output.txt
2 4 +LOR XR+R	-1
3 3 RLR LRL RLR	20
6 5 +RL+L R+OXL RLR+R RR+LX +XLLR +ROLL	18

## Spiegazione

Nel **primo caso di esempio**, il percorso seguito è questo (seguendo i colori rosso, blu, verde, porpora, grigio):



Nel **secondo caso di esempio**, il percorso seguito è questo (seguendo i colori rosso, blu, verde, porpora, grigio):





Nel **terzo caso di esempio**, il percorso seguito è questo:

+	R	L	+	L
R	+	O	X	L
R	L	R	+	R
R	R	+	L	X
+	X	L	L	R
+	R	O	L	L