

## Parco avventura (carrucole)

Giorgio e William vorrebbero andare a visitare il più grande parco avventura del mondo, composto da ben  $N$  alberi collegati tra loro da  $M$  tiri di carrucole (ogni tiro di carrucola collega una coppia di alberi). Purtroppo, un violento Tsunami si è abbattuto sul parco, abbattendo alcuni degli alberi. Naturalmente, per ogni albero abbattuto si sono persi tutti i tiri di carrucola che partivano da quell'albero. Giorgio e William si chiedono quindi se valga ancora la pena di pagare l'esoso biglietto di ingresso del parco, visto lo stato in cui si trova al momento. Aiutali calcolando quanti tiri di carrucole sono ancora rimasti in piedi!

## Implementazione

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

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

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

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

In cui:

- L'intero  $N$  rappresenta il numero (iniziale) di alberi.
- L'intero  $M$  rappresenta il numero (iniziale) di tiri di carrucole.
- L'array  $B$ , indicizzato da  $0$  a  $N - 1$ , contiene per ogni albero  $0$  se l'albero è stato abbattuto oppure  $1$  se l'albero è ancora in piedi.
- Gli array  $da$  e  $a$ , indicizzati da  $0$  a  $M - 1$ , descrivono i tiri di carrucola (che per ogni  $i$  collegano  $da[i]$  e  $a[i]$ ).
- La funzione dovrà restituire il numero di tiri di carrucola ancora rimasti in piedi, che verrà stampato sul file di output.

## Dati di input

Il file `input.txt` è composto da  $M+2$  righe. La prima riga contiene i due interi  $N$ ,  $M$ . La seconda riga contiene gli  $N$  interi  $B_i$  separati da uno spazio. Le successive  $M$  righe contengono ciascuna i due interi  $da[i]$  e  $a[i]$ .

## Dati di output

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

## Assunzioni

- $2 \leq N \leq 10\,000$ .
- $1 \leq M \leq 100\,000$ .
- $0 \leq B_i \leq 1$  per ogni  $i = 0 \dots N - 1$ .
- $0 \leq da_i, a_i \leq N - 1$  per ogni  $i = 0 \dots M - 1$ .
- Ci possono essere più tiri di carrucole tra la stessa coppia di alberi, e anche tiri di carrucole all'interno dello stesso albero.

## 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]:**  $N, M \leq 50$ .
- **Subtask 3 [40 punti]:**  $N, M \leq 1000$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
4 6 0 1 1 1 0 1 2 0 3 0 1 2 1 3 3 2	3
3 1 1 0 1 0 2	1

## Spiegazione

Nel **primo caso di esempio**, rimangono tre tiri di carrucola: tra gli alberi 1-2, 2-3 e 1-3.

Nel **secondo caso di esempio**, l'unico tiro di carrucola non è stato abbattuto.

## Hashtag Yoda (yoda)

Giorgio è andato a vedere l'ultimo episodio di Star Wars uscito da poco nelle sale cinematografiche. Essendo un grande appassionato della saga, ha intenzione di commemorare in grande stile l'uscita di tale episodio. Per farlo, ha intenzione di:


- Sostituire la propria immagine di profilo su Facebook con una foto del maestro Yoda.
- Scrivere qualsiasi cosa “in stile Yoda”, compresi: stati, messaggi, chat, e così via.

Il primo punto è facile, ma il secondo decisamente no. Giorgio deve infatti modificare ogni singola frase prima di premere Invio, e questo può alla lunga risultare stressante. Aiutalo scrivendo un programma che trasformi una frase nella relativa frase “Yodizzata”.

Ad esempio, se la frase fosse: “sia grande la forza in te”, il programma dovrebbe trasformarla in: “te in forza la grande sia”.

## Implementazione

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

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

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

C/C++	<code>void yodizza(char S[], char Y[]);</code>
Pascal	<code>procedure yodizza(var S, Y: array of char);</code>

In cui:

- L'array `S` è la stringa da yodizzare.
- Nel caso di C/C++, gli array saranno terminati con un carattere terminatore, quindi è possibile calcolarne la lunghezza con la funzione di libreria `strlen`. **Attenzione:** tenete a mente che la funzione `strlen` scorre tutta la stringa, quindi ha complessità lineare.
- Nel caso di Pascal, si può calcolare la lunghezza degli array usando la funzione di libreria `length`.
- La funzione dovrà scrivere la versione “yodizzata” della frase nell'array `Y`.

## Dati di input

Il file `input.txt` è composto da un'unica riga contenente la frase `S`.

## Dati di output

Il file `output.txt` è composto da un'unica riga contenente la frase `Y`.

## Assunzioni

- $1 \leq |S| \leq 100\,000$ , dove  $|S|$  è la lunghezza di `S`.



## 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]:**  $|S| \leq 10$ .
- **Subtask 3 [40 punti]:**  $|S| \leq 100$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
sia grande la forza in te	te in forza la grande sia
ciao	ciao

## Numero della cabala (cabala)

Giorgio vuole entrare nella setta *Oli-3*, in cui si studiano numeri occulti e misteriosi con cui dominare il mondo. Per essere ammesso, gli viene richiesto di provare il suo valore individuando il *numero della cabala*  $C$ . Dopo lunghe e difficoltose ricerche, Giorgio è riuscito a scoprire che questo numero:

- ha al più  $N$  cifre;
- tutte le sue cifre sono multiple di 3 (ma nessuna è zero);
- non vi sono due cifre adiacenti uguali;
- il resto di  $C$  modulo un certo numero  $M$  è più grande possibile.

Aiuta Giorgio a trovare il numero della Cabala ed entrare quindi nella agognata setta!

## Implementazione

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

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

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

C/C++	<code>long long occulta(int N, int M);</code>
Pascal	<code>function occulta(N, M: longint): int64;</code>

In cui:

- L'intero  $N$  rappresenta il numero di cifre di  $C$ .
- L'intero  $M$  rappresenta il modulo.
- La funzione dovrà restituire il *massimo resto* per un numero  $C$  che rispetta i vincoli nel testo modulo  $M$ , che verrà stampato sul file di output.

## Dati di input

Il file `input.txt` è composto da  $T + 1$  righe. La prima riga contiene l'unico intero  $T$ , il numero di test a cui il tuo programma dovrà rispondere. Le successive  $T$  righe contengono ciascuna due interi  $N$  ed  $M$  separati da uno spazio.

## Dati di output

Il file `output.txt` è composto da un'unica riga contenente  $T$  interi, le risposte ai test nell'ordine in cui sono stati presentati.



## Assunzioni

- $1 \leq T \leq 50$ .
- $1 \leq N \leq 18$  per ogni test.
- $2 \leq M \leq 10^9$  per ogni test.

## 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]:**  $N \leq 5$ .
- **Subtask 3 [40 punti]:**  $N \leq 11$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
1 1 7	6
3 2 68 3 1000 4 32	63 969 31

## Spiegazione

Nel **primo caso di esempio**, viene richiesto un solo test in cui il numero della cabala è 6.

Nel **secondo caso di esempio**, nel primo test il numero della cabala è 63 perché 66 ha due cifre uguali consecutive, per lo stesso motivo nel secondo test è 969, mentre nel terzo test il resto 31 può essere ottenuto per esempio con il numero della cabala 639.

## Montagne russe (rollercoaster)

William è un accanito sostenitore del videogioco *Rollercoaster Typhoon*, e passa la maggior parte del suo tempo a progettare percorsi di montagne russe. Purtroppo, la sfrenata fantasia di William si scontra con le leggi fisiche che un percorso di montagne russe deve soddisfare.

Un percorso siffatto è individuato da una sequenza  $H_i$  di  $N$  valori interi, che rappresentano l'altezza dei piloni che sorreggono la montagna russa, nell'ordine in cui vengono connessi dalle rotaie. Se un insieme consecutivo di tre o più di questi valori è in progressione aritmetica strettamente crescente (cioè la differenza tra uno di questi valori e il suo precedente è positiva e costante), allora il tratto corrispondente del percorso viene automaticamente *motorizzato*. Quando William lancia un test sul suo percorso, un veicolo di prova inizia a percorrerlo da  $i = 0$  in avanti, secondo queste regole:

1. Se il veicolo raggiunge un tratto motorizzato, viene agganciato e lo risale fino alla fine, e a quel punto viene sganciato. Se il pilone seguente si trova ad altezza strettamente inferiore dell'ultimo pilone raggiunto, il veicolo inizia una *caduta libera* come nel punto 2. Altrimenti, il veicolo perde il controllo e si verifica un incidente.
2. Se un veicolo inizia una caduta libera, prosegue nella caduta finché le altezze percorse sono *strettamente decrescenti*. Terminata una caduta libera, se il tratto seguente è motorizzato il veicolo procede come nel punto 1. Altrimenti, il veicolo inizia una *risalita inerziale* come nel punto 2.
3. Se un veicolo inizia una risalita inerziale, prosegue nella risalita finché non incontra tratti motorizzati e le altezze percorse sono *debolmente crescenti* (crescenti o costanti) e *inferiori all'altezza da cui è partita la caduta libera precedente*. Se nel punto in cui la risalita si arresta è presente un tratto motorizzato, il veicolo procede come nel punto 1. Se il pilone seguente si trova ad altezza inferiore dell'ultimo pilone raggiunto, il veicolo inizia una *caduta libera*. Altrimenti, il veicolo perde il controllo e si verifica un incidente.

Aiuta William ad evitare spiacevoli incidenti, calcolando fino a che punto il veicolo riesce a procedere correttamente!

## Implementazione

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

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

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

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

In cui:

- L'intero  $N$  rappresenta il numero di piloni della montagna russa.
- L'array  $H$ , indicizzato da  $0$  a  $N - 1$ , contiene le altezze dei piloni su cui la montagna russa è costruita.
- La funzione dovrà restituire la posizione raggiungibile da un veicolo (cioè, la fine oppure la posizione in cui avviene un incidente), che verrà stampata 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  $H_i$  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

- $1 \leq N \leq 100\,000$ .
- $1 \leq H_i \leq 100\,000$  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]:**  $N \leq 100$ .
- **Subtask 3 [40 punti]:** Non sono presenti tratti motorizzati.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

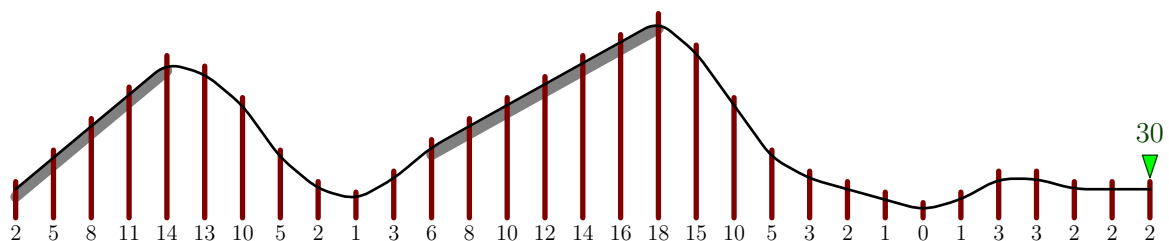
## Esempi di input/output

input.txt	output.txt
<pre>31 2 5 8 11 14 13 10 5 2 1 3 6 8 10   12 14 16 18 15 10 5 3 2 1 0 1 3     3 2 2 2</pre>	30
<pre>27 0 7 14 12 7 3 2 2 5 5 6 6 4 2 1 0   1 3 4 6 7 9 8 5 2 1 0</pre>	18
<pre>29 32 25 15 12 10 10 8 6 3 2 1 0 2 3   3 3 4 6 7 7 9 10 5 1 0 3 6 9 8</pre>	20

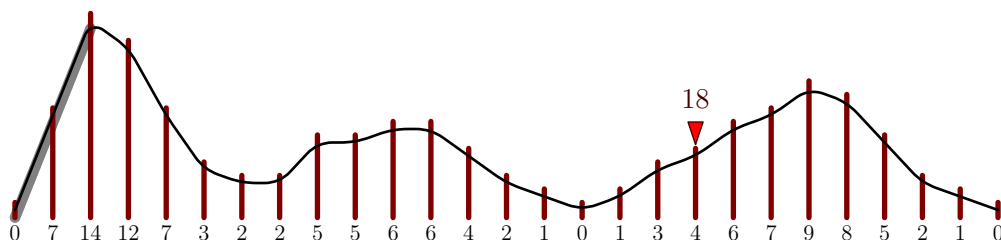


## Spiegazione

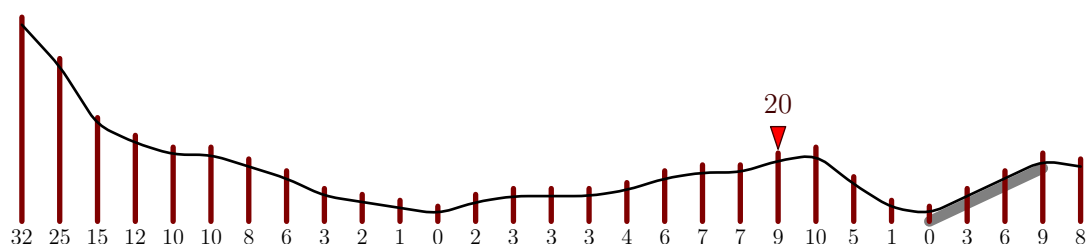
Nel **primo caso di esempio**, il veicolo è in grado di procedere fino alla fine del percorso senza incidenti.



Nel **secondo caso di esempio**, il veicolo esegue correttamente una risalita motorizzata, una caduta libera con risalita inerziale, una seconda caduta libera ma ha un incidente al durante la seconda risalita inerziale.



Nel **terzo caso di esempio**, il veicolo esegue correttamente una caduta libera, una risalita inerziale (di lunghezza uno e in piano), una seconda caduta libera ma ha un incidente durante la seconda risalita inerziale.



## Super Marco 2 (monete)

William sta di nuovo giocando a Super Marco 64, e stavolta per superare il livello deve raccogliere il maggior numero possibile di monete. Il livello è composto da una serie di piattaforme collegate tra loro. Su ciascuna piattaforma possono esserci delle monete e, appena Super Marco raggiunge una certa piattaforma, raccoglie istantaneamente tutte le monete che essa contiene.

Data la mappa del livello, e sapendo che Super Marco si trova nella piattaforma 0, determina quante monete al massimo è possibile raccogliere.

### Implementazione

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

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

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

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

In cui:

- Gli interi  $N$  ed  $M$  rappresentano rispettivamente il numero di piattaforme ed il numero di collegamenti tra di esse.
- L'array `monete`, indicizzato da 0 a  $N - 1$ , contiene il numero di monete che si trovano in ciascuna piattaforma.
- Gli array `A` e `B`, indicizzati da 0 a  $M - 1$ , identificano una coppia di piattaforme `A[i]` e `B[i]` per le quali esiste un collegamento diretto.
- La funzione dovrà restituire il massimo numero di monete che si possono raccogliere in totale. Tale numero verrà stampato sul file di output.

### Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene i due interi  $N$  e  $M$  separati da uno spazio. La seconda riga contiene gli  $N$  interi `monete[i]` separati da spazi. Ciascuna delle successive  $M$  righe contiene una coppia di interi `A[i]` e `B[i]`.

### Dati di output

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

## Assunzioni

- $1 \leq N \leq 10\,000$ .
- $0 \leq M \leq 100\,000$ .
- I collegamenti sono bidirezionali.
- $0 \leq \text{monete}[i] \leq 1000$  per ogni  $i = 0 \dots N - 1$ .
- $0 \leq A[i], B[i] \leq N - 1$ .
- $A[i] \neq B[i]$  e non ci sono collegamenti 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 [30 punti]:** Il grafo è connesso (tutte le piattaforme sono raggiungibili).
- **Subtask 3 [30 punti]:** Tutte le piattaforme hanno esattamente 1 moneta.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
5 3 1 2 1 2 1 0 2 4 1 3 0	4
1 0 1000	1000

## Spiegazione

Nel **primo caso di esempio** si può raccogliere immediatamente una moneta, dopodiché è possibile raggiungere la piattaforma 2 raccogliendo un'altra moneta, per poi tornare indietro e recarsi alla piattaforma 3 che ha ben due monete. In totale possiamo raccogliere quindi 4 monete. Purtroppo non c'è alcun modo di raggiungere le piattaforme 1 e 4.

Nel **secondo caso di esempio** c'è una sola piattaforma e non è possibile quindi spostarsi.


## Primo permissivo (primo)

William e Giorgio stanno facendo un gioco con i cosiddetti numeri *primi permissivi*. Un numero primo permissivo è un numero in cui alcune delle cifre possono essere degli asterischi. Un esempio di numero primo permissivo è  $3\star$  il quale, in base al valore che sostituiamo all'asterisco, potrebbe valere 31 oppure 37, che sono entrambi dei numeri primi. Nessun numero primo permissivo comincia per asterisco.

Aiuta i protagonisti a contare, dato un certo numero primo permissivo, a quanti numeri primi esso può corrispondere.

## Implementazione

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

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

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

C/C++	<code>int primi(char S[]);</code>
Pascal	<code>function primi(var S: string): longint;</code>

In cui:

- $S$  è una stringa che rappresenta il numero primo permissivo scelto. Per calcolarne la lunghezza si può usare la funzione `strlen` in C/C++ o la funzione `length` in Pascal.
- La funzione dovrà restituire il totale di numeri primi che possono corrispondere al numero primo permissivo dato in input. Questo totale verrà stampato sul file di output.

## Dati di input

Il file `input.txt` è composto da una sola riga che contiene la stringa  $S$ .

## Dati di output

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

## Assunzioni

- $2 \leq |S| \leq 7$ , dove  $|S|$  è la lunghezza di  $S$ .
- Il primo carattere non è un asterisco.
- C'è sempre almeno un asterisco.



## 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 [30 punti]:**  $|S| \leq 3$ .
- **Subtask 3 [30 punti]:**  $|S| \leq 5$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
3*	2
1*3	5
1**	21

## Spiegazione

Il **primo caso di esempio** corrisponde a quello nella descrizione del problema.

Nel **secondo caso di esempio** i seguenti primi vanno bene: 103, 113, 163, 173, 193.

## Acronimi forzati (acronimi)

William sta lavorando alla tesi di laurea. Durante lo sviluppo del suo progetto, ha fatto uso di una banca dati dei pagamenti degli enti pubblici italiani. Questa banca dati viene preparata e pubblicata dal SIOPE (*Sistema informativo sulle operazioni degli enti pubblici*).

Non appena ha letto l'acronimo SIOPE, William ha istintivamente cercato di associarlo alle parole che compongono la frase “Sistema informativo sulle operazioni degli enti pubblici” ma, con grande delusione, ha notato che qualsiasi associazione risultava piuttosto forzata.

Un modo per mappare l'acronimo alla frase è questo: Sistema Informativo sulle OPErazioni degli enti pubblici. Tuttavia, ci sono altri modi:

- Sistema InformativO sulle oPerazioni degli Enti pubblici.
- siStema InformativO sulle oPerazioni degli Enti pubblici.
- siStema informatIvO sulle oPerazioni degli Enti pubblici.
- ...

In tutto, William ha contato 39 modi diversi di interpretare questo acronimo (come si può intuire, è un tipo che si distrae facilmente). Mancano poche ore alla scadenza per la consegna della tesi, ma è comunque molto più importante che il seguente problema venga risolto: in quanti modi si può mappare un certo acronimo su una certa frase? Aiutalo a risolvere il problema, così che possa finalmente concentrarsi e completare la tesi.

## Implementazione

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

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

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

C/C++	<code>int acronimi(char A[], char S[]);</code>
Pascal	<code>function acronimi(var A, S: string): longint;</code>

In cui:

- $A$  e  $S$  sono, rispettivamente, l'acronimo e la stringa su cui mapparla.
- Nel caso di C/C++, gli array saranno terminati con un carattere terminatore, quindi è possibile calcolarne la lunghezza con la funzione di libreria `strlen`. **Attenzione:** tenete a mente che la funzione `strlen` scorre tutta la stringa, quindi ha complessità lineare.
- Nel caso di Pascal, gli array saranno di tipo `string`. Potete calcolarne la lunghezza con la funzione di libreria `length`.
- La funzione deve restituire il numero di modi in cui l'acronimo si può mappare nella stringa. Dal momento che questo numero può essere molto grande, è necessario restituire soltanto il resto della divisione per 1 000 000 007.

☞ Per eseguire l'operazione di modulo si può utilizzare l'operatore `%` del C/C++. In Pascal invece esiste l'operatore `mod`.

Ad esempio, il resto della divisione di 5 per 3 si calcola come `5 % 3` in C/C++ e come `5 mod 3` in Pascal. In entrambi i casi il risultato sarà 2.

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

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

## Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene l'acronimo  $A$ . La seconda riga contiene la stringa  $S$  su cui mappare l'acronimo. Entrambe le stringhe non conterranno né spazi né punteggiatura, saranno composte **solo da lettere minuscole**.

## Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema (modulo 1 000 000 007).

## Assunzioni

- $1 \leq |A| \leq 100$ , dove  $|A|$  è la lunghezza della stringa  $A$ .
- $1 \leq |S| \leq 100\,000$ , dove  $|S|$  è la lunghezza della stringa  $S$ .

## 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]:**  $|A| = 1$ .
- **Subtask 3 [45 punti]:**  $|A| \leq 5$ ,  $|S| \leq 50$ .
- **Subtask 4 [25 punti]:** Nessuna limitazione specifica.



## Esempi di input/output

input.txt	output.txt
abcd abbeceDario	2
aaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	518731105
siope sistemainformativosulleoperazionideglientipubblici	39

## Spiegazione

Nel **primo caso di esempio** l'acronimo abcd si può mappare in ABbeCeDario oppure in AbBeCeDario.

Nel **secondo caso di esempio** ci sono 25 518 731 280 modi di mappare l'acronimo ma, dal momento che dobbiamo stampare il risultato modulo 1 000 000 007, il risultato finale sarà 518 731 105.