Congettura di Bicollatz (`bicollatz`)

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

Dopo lunghi e accurati studi sulla congettura di *Lollatz*, Giorgio ha deciso che è giunto finalmente il momento di creare una nuova congettura egli stesso. Propone quindi la nuova congettura di *Bicollatz*, descritta dal seguente procedimento. Dati due numeri interi positivi A e B, si ripete il seguente passaggio:

- se A e B sono entrambi pari, si dividono entrambi per due;
- se A e B sono entrambi dispari, ciascuno dei due si moltiplica per tre e si aggiunge uno;
- altrimenti, si aggiunge 3 a quello dei due che è dispari.

La congettura dice che ripetendo questo passaggio, si arriva sempre prima o poi a ottenere che $A, B = (1, 1)$. Aiuta Giorgio a verificare la sua congettura!

Implementazione

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

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

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

C/C++	`int bicollatz(int A, int B);`
Pascal	`function bicollatz(A, B: longint): longint;`

In cui:

- Gli interi A e B rappresentano gli interi positivi iniziali.
- La funzione dovrà restituire il numero di passaggi prima di ottenere $(1, 1)$, oppure -1 se la congettura è falsa per A,B, e questo valore verrà stampato sul file di output.

Dati di input

Il file `input.txt` è composto da un'unica riga contenente i due interi A e B.

Dati di output

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

Assunzioni

- $1 \leq A, B \leq 1\,000\,000$.

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**]: $N \leq 100$.
- **Subtask 3** [**40 punti**]: $N \leq 10\,000$.
- **Subtask 4** [**30 punti**]: Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
1 1	0
14 1	10

Spiegazione

Nel **primo caso di esempio**, $A, B = (1, 1)$ senza dover effettuare alcun passaggio.

Nel **secondo caso di esempio**, si ottiene la seguente sequenza: $(14, 1)$, $(14, 4)$, $(7, 2)$, $(10, 2)$, $(5, 1)$, $(16, 4)$, $(8, 2)$, $(4, 1)$, $(4, 4)$, $(2, 2)$, $(1, 1)$.

Las Vegas (`insegna`)

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

Giorgio e William si devono incontrare segretamente in un locale di Las Vegas per definire i problemi delle gare a squadre di quest'anno. Purtroppo, a Las Vegas tutti i locali hanno insegne circolari rotanti, quindi è molto difficile trovare il luogo sul quale si sono accordati.

Dopo un po' di tempo passato a cercare William, Giorgio legge una stringa G su un'insegna e decide di contattarlo. Quest'ultimo nel frattempo si è perso per le strade di Las Vegas, ma per fortuna riesce anche lui a vedere un'insegna con su scritta una stringa W. Dal momento che i tempi per le olimpiadi si stanno facendo stringenti, i due vogliono capire se almeno si trovano nella stessa parte della città, così da riuscire ad incontrarsi e decidere una volta per tutte i problemi.

Le due stringhe G e W si riferiscono allo stesso luogo quando sono la stessa stringa *a meno di una permutazione ciclica* (o rotazione). Indichiamo con $|G|$ la lunghezza di G. Se è possibile spezzare la stringa G in un punto i compreso tra 0 e $|G| - 1$, ed è possibile poi ottenere una stringa uguale a W scambiando di posizione le due "metà" di G prodotte, allora diremo che G è una permutazione ciclica di W (e viceversa).

Aiuta Giorgio e William a capire se le due stringhe sono la stessa insegna!

Implementazione

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

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

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

C/C++	`int confronta(int N, char* G[], char* W[]);`
Pascal	`function confronta(N: longint; var G, W: array of char): longint;`

In cui:

- L'intero N rappresenta la lunghezza delle due stringhe.
- Gli array G e W rappresentano le stringhe viste da Giorgio e da William, rispettivamente.
- La funzione dovrà restituire 1 se le due stringhe si riferiscono alla stessa insegna, altrimenti 0.

Dati di input

Il file `input.txt` è composto da tre righe. La prima riga contiene l'unico intero N. La seconda riga contiene la stringa G. La terza riga contiene la stringa W.

Dati di output

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

Assunzioni

- $1 \leq N \leq 5000$.
- Le due stringhe sono composte da caratteri compresi tra `a` e `z`.

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
5 abcde fghij	0
7 abcdefg defgabc	1

Spiegazione

Nel **primo caso di esempio** è chiaramente impossibile ottenere G da W o viceversa.

Nel **secondo caso di esempio**, invece, basta spezzare G nel punto $i = 3$ ottenendo le due metà `abc` e `defg` che possiamo scambiare per ottenere W.

Spreco di energia (`energia`)

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

William è molto attento agli sprechi in casa: ogni volta che una spia o un display rimangono accesi, lui corre a spegnerli per risparmiare sulla bolletta. In certi casi è arrivato anche a scollegare i fili dei display meno essenziali (quello del videoregistratore, ad esempio) pur di evitare di lasciarli perennemente accesi. Purtroppo però, il suo *contatore elettrico* è stato installato dal fornitore nazionale di energia, e sarebbe illegale manometterlo al fine di spegnergli il display! Il povero William quindi non può far altro che accettare di pagare una bolletta esorbitante a causa di questo display costantemente acceso.

Il contatore installato a casa di William presenta un *display a sette segmenti*, lo stesso che si trova in molti tipi di radiosveglia. Questo tipo di display dispone di 7 led per ciascuna cifra.

Nello schema di seguito si può vedere il numero di led richiesti per illuminare le varie cifre. Possiamo notare, ad esempio, che la cifra `1` richiede l'accensione di 2 soli led, mentre la cifra `8` richiede che vengano accesi tutti e 7:

All'inizio del mese il contatore verrà resettato a 0 e, crescendo di 1 alla volta a velocità costante, raggiungerà il valore K alla fine del mese. Diciamo quindi che l'*energia totale* consumata dal display è pari alla *somma* dell'energia consumata per illuminare i numeri da 0 a K. Per comodità, diremo inoltre che l'energia consumata per illuminare un singolo numero è pari al numero di led che è necessario illuminare per ciascuna delle sue cifre.

Ad esempio, per illuminare il solo numero `1234` servono $2 + 5 + 5 + 4 = 16$ unità di energia. William intende ora protestare presso l'agenzia elettrica per questo inaccettabile spreco, e vuole quindi sapere la somma delle unità consumate durante tutto il mese. Aiutalo scrivendo un programma che calcoli quanta energia verrà consumata per illuminare il display del contatore per tutta la durata del mese!

Implementazione

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

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

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

C/C++	`long long int energia(long long int K);`
Pascal	`function energia(K: int64): int64;`

In cui:

- L'intero K rappresenta il valore che sarà scritto sul contatore alla fine del mese.
- La funzione dovrà restituire il numero di unità di energia consumate in totale dal display del contatore elettrico durante tutto il mese.

Dati di input

Il file `input.txt` è composto da un'unica riga contenente l'unico intero K.

Dati di output

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

Assunzioni

- $0 \leq K \leq 1\,000\,000\,000\,000\,000$ (10^{15}).
- Non vengono illuminate cifre `0` non significative.

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]**: $K \leq 100$.
- **Subtask 3 [40 punti]**: $K \leq 1\,000\,000$.
- **Subtask 4 [20 punti]**: Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
3	18
10	57
123456789	4800207171

Spiegazione

Nel **primo caso di esempio**, il contatore illuminerà i seguenti numeri:

- `0` (6 unità di energia)
- `1` (2 unità di energia)
- `2` (5 unità di energia)
- `3` (5 unità di energia)

Nel **secondo caso di esempio**, il contatore illuminerà i seguenti numeri:

- `0` (6 unità di energia)
- `1` (2 unità di energia)
- `2` (5 unità di energia)
- `3` (5 unità di energia)
- `4` (4 unità di energia)
- `5` (5 unità di energia)
- `6` (6 unità di energia)
- `7` (3 unità di energia)
- `8` (7 unità di energia)
- `9` (6 unità di energia)
- `10` ($2 + 6 = 8$ unità)

Permutazione genetica (`permutazione`)

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

Da quando ha cominciato a lavorare come ricercatore, Giorgio si diverte a fare esperimenti su cavie animali. In particolare si è deciso ad ottenere due topi superintelligenti, proprio come nel suo cartone preferito: per raggiungere questo obiettivo però, Giorgio deve incrementare esponenzialmente il potenziale genetico dei due topi da laboratorio a sua disposizione. Il nostro protagonista ritiene infatti che una semplice mutazione genetica non sia sufficiente a raggiungere l'obiettivo, e per questo motivo decide di produrre una *permutazione genetica*.

Per gli scopi di Giorgio, definiamo una *permutazione genetica* come una stringa lunga 26 caratteri che vanno dalla lettera `A` alla `Z`. In particolare, affinché la mutazione abbia effetto, è necessario che la sua *funzione espotenziale* valga esattamente K.

Per calcolare la funzione espotenziale di una permutazione genetica dobbiamo osservare le variazioni tra le 25 coppie di lettere consecutive (le prime due lettere, le seconde due lettere, e così via). Ad esempio, consideriamo la permutazione genetica banale:

```
A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
```

Le variazioni tra le coppie di lettere consecutive sono tutte di una sola lettera (infatti la distanza tra `A` e `B`, tra `B` e `C` e così via, è sempre pari a 1). Prendiamo l'insieme di tutte le variazioni, cancelliamo i duplicati, ed il numero di valori rimanenti sarà pari alla funzione espotenziale della permutazione!

Lo stesso vale per quest'altra permutazione, dato che la distanza tra `Z` e `Y` è ancora pari ad 1:

```
Z, Y, X, W, V, U, T, S, R, Q, P, O, N, M, L, K, J, I, H, G, F, E, D, C, B, A
```

Un esempio più significativo permette di capire meglio:

```
P, L, T, A, V, H, F, W, D, Z, J, O, S, G, U, K, X, R, I, N, C, Y, E, Q, M, B
```

In questo caso le distanze tra le lettere consecutive sono:

$$4, 8, 19, 21, 14, 2, 17, 19, 22, 16, 5, 4, 12, 14, 10, 13, 6, 9, 5, 11, 22, 20, 12, 4, 11$$

Eliminando i duplicati otteniamo:

$$2, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 19, 20, 21, 22$$

Essendoci 17 valori diversi, la funzione espotenziale vale 17.

Scrivi un programma che, ricevuto K, produca una qualsiasi permutazione genetica che abbia funzione espotenziale pari a K.

Implementazione

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

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

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

C/C++	`void permuta(int K, char* P[]);`
Pascal	`procedure permuta(K: longint; var P: array of char);`

In cui:

- L'intero K è il valore della funzione espotenziale richiesto.
- L'array di caratteri `P` andrà riempito dalla funzione con 26 caratteri (indicizzati da 0 a 25) che formano una permutazione genetica corretta.

Dati di input

Il file `input.txt` è composto da un'unica riga contenente l'unico intero K.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente una stringa di 26 caratteri, la risposta a questo problema.

Assunzioni

- $1 \leq K \leq 25$.

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]**: $K \leq 3$.
- **Subtask 3 [30 punti]**: $K \geq 23$.
- **Subtask 4 [40 punti]**: Nessuna limitazione specifica.

Esempi di input/output

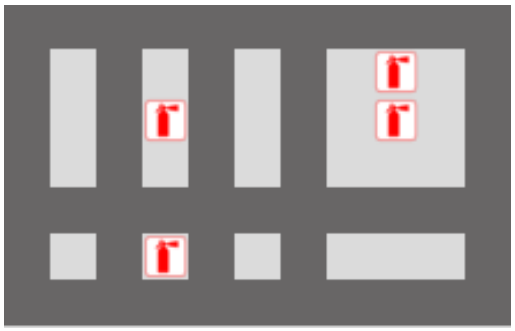
input.txt	output.txt
1	ABCDEFGHIJKLMNOPQRSTUVWXYZ
1	ZYXWVUTSRQPONMLKJIHGFEDCBA
17	PLTAVHFWDZJOSGUKXRINCYEQMB

Controllo degli estintori (`estintori`)

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

William e Giorgio sono appena stati promossi a ispettori di sicurezza alla Scuola Internazionale di Alta Formazione (SIAF), la struttura che si trova a Volterra e che viene utilizzata per svolgere molti degli stage di allenamento dei Probabili Olimpici. Affinché la struttura possa essere adatta ad ospitare tanti giovani ragazzi e ragazze, è necessario che le norme di sicurezza vengano rispettate alla lettera. Il direttore della SIAF ha quindi chiesto ai nuovi ispettori di aiutarlo ad analizzare la piantina della struttura in modo da trovare eventuali anomalie.

La richiesta del direttore è la seguente: viene data una mappa della SIAF in formato testuale, in cui le stanze (carattere `.`) sono delimitate dai muri (carattere `#`) e possono contenere 0 o più estintori (carattere `@`), ed è necessario capire in quali stanze *non c'è* un estintore. Ad esempio, nella seguente mappa possiamo notare che ci sono 5 stanze senza estintori:



```
##########
#.#.#.#.@.#
#.#@#.#.@.#
#.#.#.#...#
##########
#.#@#.#...#
##########
```

I muri della SIAF formano sempre una griglia, per cui le stanze sono tutte rettangolari. Quante stanze sono senza estintori?

Implementazione

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

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

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

C/C++	`int mancanti(int R, int C, char** M);`
Pascal	`function mancanti(R, C: longint; var M: matrix): longint;`

In cui:

- Gli interi R e C rappresentano il numero di righe e colonne della mappa, rispettivamente.
- La matrice `M`, indicizzata da 0 a $R-1$ sulle righe e da 0 a $C-1$ sulle colonne, contiene i caratteri che descrivono la piantina dell'edificio.
- La funzione dovrà restituire il numero di stanze senza estintore, che verrà poi stampato sul file di output.

Dati di input

Il file `input.txt` è composto da $R + 1$ righe. La prima riga contiene due interi R e C. Ognuna delle successive R righe è lunga C caratteri e descrive una riga della mappa.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, il numero di stanze senza estintore.

Assunzioni

- $3 \leq R, C \leq 1000$.
- Ogni stanza è sempre composta da almeno un carattere `.` o un carattere `@`. In altre parole, non ci sono stanze "schiacciate" tra due muri adiacenti.

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**]: $R, C \leq 30$.
- **Subtask 3** [**40 punti**]: Tutte le stanze hanno zero estintori.
- **Subtask 4** [**30 punti**]: Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
4 7 ####### #..#..# #@.#..# #######	1
7 11 ########### #.#.#.#.@.# #.#@#.#.@.# #.#.#.#...# ########### #.#@#.#...# ###########	5

Spiegazione

Il **secondo caso di esempio** è lo stesso input che si trova nella figura nel testo.

Maxi-server (`ricostruzioni`)

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

Ora che le gare a squadre di informatica stanno coinvolgendo sempre più la nazione, il quartier generale delle Olimpiadi ha bisogno di costruire un nuovo maxi-server in modo da gestire efficientemente la competizione. Questo maxi-server dovrà essere lungo K metri, e dovrà essere collocato in un punto segreto in cui però la corrente elettrica sia facilmente disponibile. Per questo motivo William ha selezionato, tra tutte le linee elettriche italiane, la linea *A/C51* che attraversa le alpi da una parte all'altra garantendo la giusta dose di riservatezza. Ora rimane solo da individuare il punto esatto in cui costruire il maxi-server!

A complicare la scelta, tuttavia, rimane il fatto che la linea *A/C51* attraversa zone con altitudini molto diverse, e che per giunta cambiano molto rapidamente, mentre il maxi-server andrà costruito necessariamente in piano. William sa che nel metro i della linea (lunga complessivamente N metri) l'altitudine sul livello del mare è di A_i metri. Sa inoltre che una volta scelto un intervallo di K metri all'interno della linea, il costo per spianare quell'intervallo sarà proporzionale alla *massima differenza di altitudine* presente in quell'intervallo, vale a dire la differenza tra l'altitudine più alta e quella più bassa.

Aiuta William a trovare il punto più pianeggiante della linea *A/C51*!

Implementazione

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

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

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

C/C++	`int spiana(int N, int K, int A[]);`
Pascal	`function spiana(N, K: longint; var A: array of longint): longint;`

In cui:

- L'intero N rappresenta la lunghezza totale in metri della linea *A/C51*.
- L'intero K rappresenta la lunghezza in metri del maxi-server da costruire.
- L'array `A`, indicizzato da 0 a $N-1$, contiene le altitudini dei vari metri della linea.
- La funzione dovrà restituire la minima differenza di altitudine per un intervallo di K metri, che verrà stampata 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 K. La seconda riga contiene gli N interi A_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

- $1 \leq K \leq N \leq 100\,000$.
- $0 \leq A_i \leq 1\,000\,000$ per ogni $i = 0 \ldots N-1$.

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** [**30 punti**]: $N \leq 100$.
- **Subtask 3** [**20 punti**]: $K \leq 100$.
- **Subtask 4** [**20 punti**]: $A_i \leq 100$ per ogni $i = 0 \ldots N-1$.
- **Subtask 5** [**20 punti**]: Nessuna limitazione specifica.

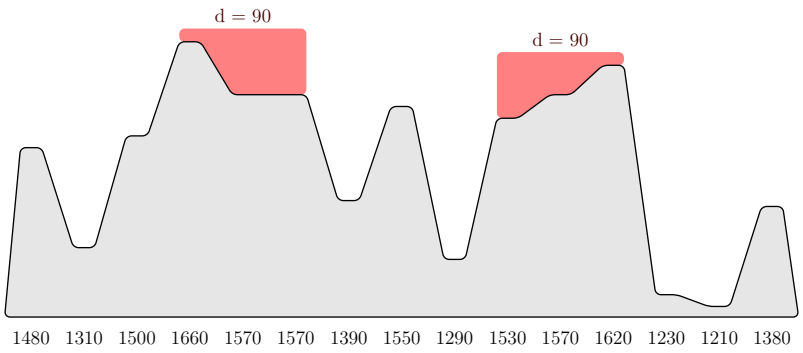
Esempi di input/output

Esempi di input/output

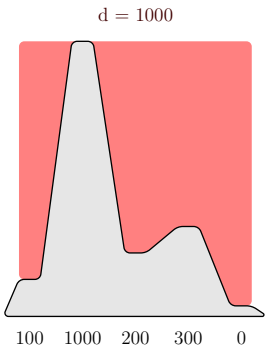
input.txt	output.txt
15 3 1480 1310 1500 1660 1570 1570 1390 1550 1290 1530 1570 1620 1230 1210 1380	90
5 5 100 1000 200 300 0	1000
10 5 6870 7829 8708 6060 1918 1956 5344 3666 3850 5899	3426

Spiegazione

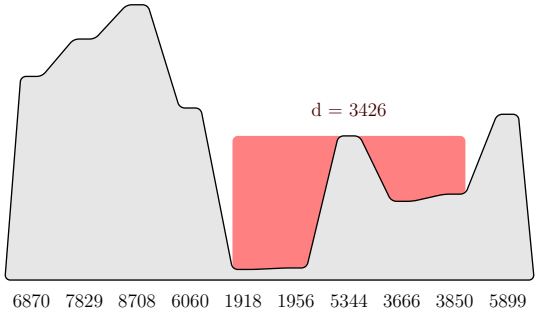
Nel **primo caso di esempio**, si può ottenere una differenza di 90 sia selezionando le altitudini 1660 – 1570 – 1570 che le altitudini 1530 – 1570 – 1620.



Nel **secondo caso di esempio**, siamo costretti a selezionare tutta quanta la lunghezza della linea per una differenza di altitudine di 1000.



Nel **terzo caso di esempio**, si può ottenere una differenza di 3426 selezionando le altitudini 1918 – 1956 – 5344 – 3666 – 3850.



Genealogia antica (olimpo)

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

La passione di Giorgio per i giochi di ruolo lo ha portato di recente a interessarsi alla mitologia classica. Sta quindi cercando di ricostruire la lunga genealogia dei discendenti di Zeus, nella speranza di dimostrare che la sua stirpe discende proprio dalla somma divinità. Purtroppo però le informazioni che riesce a carpire dagli antichi manoscritti sono molto carenti, e non è facile ricostruire l'albero dei discendenti di Zeus!

In questo albero, Zeus (indicato con il numero 0) costituisce la radice, i suoi figli costituiscono i suoi immediati successori, e così via fino a elencare tutta la genealogia. Di ognuno degli $N-1$ discendenti di Zeus si conosce o il padre (`P`), oppure il nonno (`N`) oppure il bisnonno (`B`); ma sempre solo uno dei tre.

Dopo una lunga ricerca, Giorgio non è riuscito a trovare prove del suo collegamento con Zeus. L'unica soluzione è quindi falsificare un qualche atto di nascita e rimediare alla sfortunata circostanza! Per rendere il falso più credibile possibile, Giorgio ha deciso di selezionare come suo antenato *il discendente di Zeus più lontano* all'interno dell'albero genealogico che ha ricostruito. Quanto è distante questo discendente da Zeus?

Implementazione

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

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

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

C/C++	`int falsifica(int N, char Q[], int A[]);`
Pascal	`function falsifica(N: longint; var Q: array of char; var A: array of longint): longint;`

In cui:

- L'intero N rappresenta il numero di discendenti di Zeus individuati da Giorgio.
- L'array `Q`, indicizzato da 0 a $N-1$, specifica per ogni discendente i se ciò che si conosce è il padre (`'P'`), il nonno (`'N'`) o il bisnonno (`'B'`).
- L'array `A`, indicizzato da 0 a $N-1$, specifica per ogni discendente i l'indice `A[`i`]` del suo antenato conosciuto.
- La funzione dovrà restituire la distanza tra Zeus e il suo discendente più lontano, che verrà stampata sul file di output.

Dati di input

Il file `input.txt` è composto da N righe. La prima riga contiene l'unico intero N, mentre la riga i-esima contiene `Q[`i`]`, `A[`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

- $1 \leq N \leq 1\,000\,000$.
- $0 \leq A_i \leq N-1$ per ogni $i = 1 \ldots N-1$, e $A_0 = -1$.
- Q_i è tra `'P'`, `'N'`, `'B'` per ogni $i = 1 \ldots N-1$, e Q_0 = `'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.

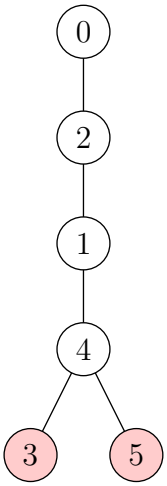
- **Subtask 1** [**10 punti**]: Casi d'esempio.
- **Subtask 2** [**20 punti**]: $N \leq 10$.
- **Subtask 3** [**40 punti**]: $N \leq 1000$.
- **Subtask 4** [**30 punti**]: Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
6 N 0 P 0 P 4 B 0 P 4	4
10 P 0 B 0 P 0 P 0 N 0 N 0 N 1 N 3 P 2	4

Spiegazione

Nel **primo caso di esempio**, i discendenti più lontani sono i numeri 3 e 5 che hanno come antenati 4, 1, 2, 0 e quindi sono a distanza 4 da Zeus.



Nel **secondo caso di esempio**, il discendente più lontano è il numero 9. Anche se non si possono ricostruire esattamente tutti i suoi antenati, il suo padre è 2 che ha come bisnonno Zeus, quindi è ancora a distanza 4 dal mitico progenitore.