Problema A. Țelină

Fişier de intrare:	`telina.in`
Fişier de ieşire:	`telina.out`
Limită de timp:	0.5 secunde
Limită de memorie:	16 megabytes

Ţelina, sau apium graveolens, este o plantă din familia apiaceae. Poate atinge o înălţime de până la 1 metru. Frunzele sale sunt mari, penat-lobate. Florile sunt mici, de obicei având culoarea albă. Fructul său este achenă. Poate rezista până la temperaturi aproape de 0 grade Celsius. Perioada de înflorire este la începutul toamnei. Este o plantă hidrofilă. [sursă: wikipedia.ro]

Se dă un şir de N caractere $S_1S_2 \dots S_N$. Ţelina citeşte (poate citi!) acest şir din dreapta spre stânga, începând cu caracterul de pe poziţia N şi terminând cu cel de pe poziţia 1.

Pe măsură ce aceasta parcurge caractere, le introduce într-o stivă T (iniţial vidă). În momentul în care citeşte caracterul A_{N-i+1} (la pasul i) îl introduce în vârful stivei T. Inainte de citi caracterul de la pasul $i+1$ ţelina poate inversa toate caracterele din stivă sau continuă la pasul următor.

Ţelina vă roagă să aflaţi şirul minim lexicografic ce se poate forma în stiva T (primul element fiind in vârful stivei iar ultimul element în capătul stivei) la finalizarea operaţiilor de la pasul N.

Date de intrare

În fişierul de intrare *telina.in* se află pe prima linie şirul S.

Date de ieșire

În fişierul de ieşire *telina.out* va conţine pe prima linie şirul minim lexicografic ce se poate obţine în T aplicând operaţiile descrise anterior.

Restricții

$1 \leq N \leq 10^6$.

S conţine doar caractere mici ale alfabetului englez.

Exemplu

telina.in	telina.out
telina	anilet

Explicație

La pasul 1, $T = a$. După citirea caracterului n stiva $T = na$. Înainte de a termina pasul 2 ţelina întoarce toate elementele din T iar acum $T = an$.

După cei 6 paşi $T = anilet$.

Problema B. Coliziune

Fişier de intrare:	`coliziune.in`
Fişier de ieşire:	`coliziune.out`
Limită de timp:	2 secunde
Limită de memorie:	4 megabytes

Agenţia de securitate naţională urmăreşte îndeaproape concursul RoTopCoder. Aceasta a descoperit că în cadrul concursului se află concurenţi care au reuşit să identifice un algoritm eficient pentru determinarea coliziunilor unei funcţii $\mathcal{H}$. O coliziune reprezintă 2 valori x, y pentru care $\mathcal{H}(x) = \mathcal{H}(y)$.

Este un lucru binecunoscut în criptologie (ştiinţă ce se ocupă cu studiul codurilor) că această problemă este una dificilă (pentru o funcţie $\mathcal{H}$ bine aleasă).

Fie o funcţie $\mathcal{H}$ ce primeste ca date intrare şiruri de lungime variabilă continând doar cifre de 0 şi 1 (şir binar). Considerăm un şir binar S de lungime N pentru care primul caracter se află pe poziţia 1 (numerotarea cifrelor incepe de la 1).

$\mathcal{H}$ este calculată în felul următor:

$$\mathcal{H}(S) = (\sum_{i=1}^{N} (S(i) + 1)B^{N-i}) \mod M \quad (1)$$

unde cu $x \mod M$ am notat restul împărţirii lui x la M.

Dându-se 2 numere naturale B şi M, se cere sa găsiţi 2 şiruri binare diferite x, y, astfel încât $\mathcal{H}(x) = \mathcal{H}(y)$.

Date de intrare

În fişierul de intrare *coliziune.in* pe prima linie se vor afla 2 numere întregi M si B (cu semnificaţia descrisă în enunţ) separate prin spaţiu. ($2 \leq B \leq M - 2$, $4 \leq M \leq 10^{14}$)

Date de ieșire

În fişierul de ieşire *coliziune.out* se vor afla 2 şiruri binare x, y separate printr-un spaţiu astfel încât $\mathcal{H}(x) = \mathcal{H}(y)$.

Exemple

coliziune.in	coliziune.out
2 1 1 2	1
3 3 1 2 3	0

Explicație

In the first example, we print 1.

In the second example, after thinking for a while, we are able to print 0.