Locks

Carl Svensson

Heidi is trying to break into the Marmots Security Agency (MSA) headquarters. She has found a back door which has a lock with a flawed password system which she is now trying to break open. The lock works by taking a password as input and comparing it to the correct password character by character. If all characters match, the password is correct and the door opens. If there is a mismatch between two characters the system aborts immediately and remains locked.

Comparing two characters takes a small amount of time. By measuring how much time the lock takes before aborting, it is possible to infer how many characters were correct and therefore eventually breaking the lock. The passwords consist of at most 100 lower and uppercase characters and are case-sensitive.

You are given a function `login` which takes a single string as argument and returns the time it took to execute the function. As you can see from the behaviour of `login`, the time per character may vary slightly. Your goal is to call the function with the correct password. If you do so, it will return -1. Your implementation of `timing()` must return the correct password. You may call `login` as many times as you wish.

Serrures

Carl Svensson

Heidi essaie d'entrer dans le quartier général de la MSA (agence de securité des marmottes). Elle a localisé une porte de derrière, verrouillée par une serrure. Heidi sait que le système de mots de passe de cette serrure a une faille. La serrure compare le mot de passé entré par Heidi à la combinaison correcte, caractère par caractère. En cas de concordance absolue, le mot de passe entré est correct et la porte s'ouvre. Les mots de passe sont comparés de la gauche vers la droite et le système s'arrête dès qu'une divergence entre deux caractères est observée, la porte restant fermée.

La comparaison de deux caractères prend un certain temps. Ainsi le temps qu'il faut à la serrure pour évaluer un mot de passe permet d'estimer le nombre de caractères corrects avant le caractère erroné. Tirez-en un avantage pour trouver le bon mot de passe pour Heidi. Les mots de passe sont formés par 100 caractères au maximum, majuscules ou minuscules et sont sensibles à la casse.

Donne en argument une chaîne de caractères à la fonction `login` et tu obtiendras en réponse le temps d'exécution de la fonction. Comme tu peux verifier avec le comportement de `login`, le temps de comparaison de deux caractères peut varier legèrement. Ton but est d'appeler la fonction avec le bon mot de passe, auquel elle répondra avec -1. Ton implementation de `timing()` doit retourner ce mot de passe. Tu peux appeler `login` aussi souvent que tu veux.