Some file managers sort filenames taking into account case of the letters, others compare strings as if all of the letters are of the same case. That may lead to different ways of filename ordering.

Call two filenames *an unstable pair* if their ordering depends on the case.

To compare two filenames a and b, find the first position i at which a[i] ≠ b[i]. If a[i] < b[i], then a < b. Otherwise a > b. If such position doesn't exist, the shorter string goes first.

Given two filenames, check whether they form an unstable pair.

**Example**

* For filename1 = "aa" and filename2 = "AAB", the output should be  
  isUnstablePair(filename1, filename2) = true.

Because "AAB" < "aa", but "AAB" > "AA".

* For filename1 = "A" and filename2 = "z", the output should be  
  isUnstablePair(filename1, filename2) = false.

Both "A" < "z" and "a" < "z".

**Input/Output**

* **[time limit] 20000ms (swift)**
* **[input] string filename1**

A non-empty string of letters and digits.

*Constraints:*  
1 ≤ filename1.length ≤ 10.

* **[input] string filename2**

A non-empty string of letters and digits. It's guaranteed that there is no ambiguity, i.e. even being considered in the same case strings are never equal.

*Constraints:*  
1 ≤ filename2.length ≤ 10.

* **[output] boolean**

true if filename1 and filename2 form an unstable pair, false otherwise.

<https://codefights.com/arcade/code-arcade/book-market/Ky2mjgmxnWLi6KNPp>

<https://github.com/Swift-Solutions/CodeFights/blob/master/solutions/is_unstable_pair.swift>

|  |
| --- |
| func isUnstablePair(filename1: String, filename2: String) -> Bool { |
|  | let f1 = filename1 |
|  | let f2 = filename2 |
|  | let f1Up = filename1.uppercaseString |
|  | let f2Up = filename2.uppercaseString |
|  | let f1Low = filename1.lowercaseString |
|  | let f2Low = filename2.lowercaseString |
|  | if isSmaller(f1, f2) { |
|  | if isSmaller(f2Low, f1Low) || f2Low == f1Low { |
|  | return true |
|  | } |
|  | if isSmaller(f2Up, f1Up) || f2Up == f1Up { |
|  | return true |
|  | } |
|  | } |
|  | if isSmaller(f2, f1) { |
|  | if isSmaller(f1Low, f2Low) || f1Low == f2Low { |
|  | return true |
|  | } |
|  | if isSmaller(f1Up, f2Up) || f1Up == f2Up { |
|  | return true |
|  | } |
|  | } |
|  | return false |
|  | } |
|  |  |
|  | func isSmaller(a: String, \_ b: String) -> Bool { |
|  | var chars1 = a.characters.map({String($0)}) |
|  | var chars2 = b.characters.map({String($0)}) |
|  | let maxChars = min(chars1.count, chars2.count) |
|  | for i in 0..<maxChars { |
|  | let isSmall1 = chars1[i] == chars1[i].lowercaseString |
|  | let isSmall2 = chars2[i] == chars2[i].lowercaseString |
|  | if (isSmall1 && isSmall2) || (!isSmall1 && !isSmall2) { |
|  | if chars1[i] < chars2[i] { |
|  | return true |
|  | } else if chars1[i] > chars2[i] { |
|  | return false |
|  | } |
|  | } else { |
|  | if isSmall2 { |
|  | return true |
|  | } else { |
|  | return false |
|  | } |
|  | } |
|  | } |
|  | return chars1.count < chars2.count |
|  | } |