1. **Fashion in Berland**

**Difficulty:**Secret   **Source:**Codeforces   **Time Limit:**1000ms   **Memory Limit:**256MB

According to rules of the Berland fashion, a jacket should be fastened by all the buttons except only one, but not necessarily it should be the last one. Also if the jacket has only one button, it should be fastened, so the jacket will not swinging open.

You are given a jacket with 𝑁*N* buttons. Determine if it is fastened in a right way.

**Input Format**

The first line contains integer 𝑁*N* (1≤𝑁≤1000)(1≤*N*≤1000) — the number of buttons on the jacket.

The second line contains 𝑁*N* integers 𝑎𝑖*a*​*i*​​ (0≤𝑎𝑖≤1)(0≤*a*​*i*​​≤1). The number 𝑎𝑖=0*a*​*i*​​=0 if the 𝑖*i*-th button is not fastened. Otherwise 𝑎𝑖=1*a*​*i*​​=1.

**Output Format**

In the only line print the word YES if the jacket is fastened in a right way. Otherwise print the word NO.

**Sample test**

|  |
| --- |
| **input** |
| 3 1 0 0 |
| **output** |
| NO |

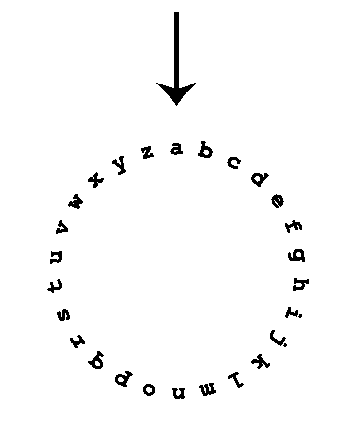
|  |
| --- |
| **input** |
| 3 1 0 1 |
| **output** |
| YES |

1. **Night at the Museum**

**Difficulty:**Secret   **Source:**Codeforces   **Time Limit:**1000ms   **Memory Limit:**256MB

Grigoriy, like the hero of one famous comedy film, found a job as a night security guard at the museum. At first night he received embosser and was to take stock of the whole exposition.

Embosser is a special devise that allows to "print" the text of a plastic tape. Text is printed sequentially, character by character. The device consists of a wheel with a lowercase English letters written in a circle, static pointer to the current letter and a button that print the chosen letter. At one move it's allowed to rotate the alphabetic wheel one step clockwise or counterclockwise. Initially, static pointer points to letter 'a'. Other letters are located as shown on the picture:



After Grigoriy add new item to the base he has to print its name on the plastic tape and attach it to the corresponding exhibit. It's not required to return the wheel to its initial position with pointer on the letter 'a'.

Our hero is afraid that some exhibits may become alive and start to attack him, so he wants to print the names as fast as possible. Help him, for the given string find the minimum number of rotations of the wheel required to print it.

**Input Format**

The only line of input contains the name of some exhibit — the non-empty string consisting of no more than 100100 characters. It's guaranteed that the string consists of only lowercase English letters.

**Output Format**

Print one integer — the minimum number of rotations of the wheel, required to print the name given in the input.

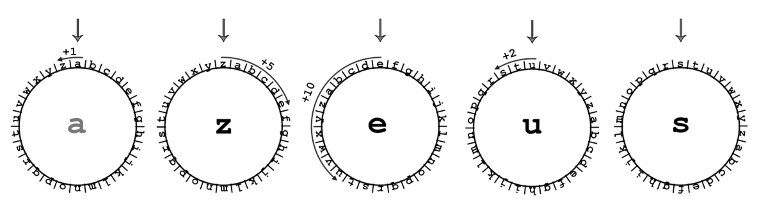
**Sample test**

|  |
| --- |
| **input** |
| ares |
| **output** |
| 34 |

|  |
| --- |
| **input** |
| zeus |
| **output** |
| 18 |

|  |
| --- |
| **input** |
| map |
| **output** |
| 35 |

**Explanation for sample test**



To print the string from the first sample it would be optimal to perform the following sequence of rotations:

1. from 'a' to 'z' (11 rotation counterclockwise),
2. from 'z' to 'e' (55 clockwise rotations),
3. from 'e' to 'u' (1010 rotations counterclockwise),
4. from 'u' to 's' (22 counterclockwise rotations). In total, 1+5+10+2=181+5+10+2=18 rotations are required.
5. **Bear and Game**

**Difficulty:**Secret   **Source:**Codeforces   **Time Limit:**2000ms   **Memory Limit:**256MB

Bear Limak likes watching sports on TV. He is going to watch a game today. The game lasts 9090 minutes and there are no breaks.

Each minute can be either interesting or boring. If 1515 consecutive minutes are boring then Limak immediately turns TV off.

You know that there will be 𝑛*n* interesting minutes 𝑡1,𝑡2,...,𝑡𝑛*t*​1​​,*t*​2​​,...,*t*​*n*​​. Your task is to calculate for how many minutes Limak will watch the game.

**Input Format**

The first line of the input contains one integer 𝑛*n* (1≤𝑛≤90)(1≤*n*≤90) — the number of interesting minutes.

The second line contains 𝑛*n* integers 𝑡1,𝑡2,...,𝑡𝑛*t*​1​​,*t*​2​​,...,*t*​*n*​​ (1≤𝑡1<𝑡2<...<𝑡𝑛≤90)(1≤*t*​1​​<*t*​2​​<...<*t*​*n*​​≤90), given in the increasing order.

**Output Format**

Print the number of minutes Limak will watch the game.

**Sample test**

|  |
| --- |
| **input** |
| 3  7 20 88 |
| **output** |
| 35 |

|  |
| --- |
| **input** |
| 9  16 20 30 40 50 60 70 80 90 |
| **output** |
| 15 |

|  |
| --- |
| **input** |
| 9  15 20 30 40 50 60 70 80 90 |
| **output**copy |
| 90 |

**Explanation for sample test**

In the first sample, minutes 21,22,...,3521,22,...,35 are all boring and thus Limak will turn TV off immediately after the 3535-th minute. So, he would watch the game for 3535 minutes.

In the second sample, the first 1515 minutes are boring.

In the third sample, there are no consecutive 1515 boring minutes. So, Limak will watch the whole game.

1. **Vitaly and Strings**

**Difficulty:**Secret   **Source:**Codeforces   **Time Limit:**1000ms   **Memory Limit:**256MB

Vitaly is a diligent student who never missed a lesson in his five years of studying in the university. He always does his homework on time and passes his exams in time.

During the last lesson the teacher has provided two strings s and t to Vitaly. The strings have the same length, they consist of lowercase English letters, string 𝑠*s* is lexicographically smaller than string 𝑡*t*. Vitaly wondered if there is such string that is lexicographically larger than string 𝑠*s* and at the same is lexicographically smaller than string 𝑡*t*. This string should also consist of lowercase English letters and have the length equal to the lengths of strings 𝑠*s* and 𝑡*t*.

Let's help Vitaly solve this easy problem!

String 𝑠=𝑠1𝑠2...𝑠𝑛‾*s*=​*s*​1​​*s*​2​​...*s*​*n*​​​​​ is said to be lexicographically smaller than 𝑡=𝑡1𝑡2...𝑡𝑛‾*t*=​*t*​1​​*t*​2​​...*t*​*n*​​​​​, if there exists such 𝑖*i*, that 𝑠1=𝑡1,𝑠2=𝑡2,...,𝑠𝑖−1=𝑡𝑖−1,𝑠𝑖<𝑡𝑖*s*​1​​=*t*​1​​,*s*​2​​=*t*​2​​,...,*s*​*i*−1​​=*t*​*i*−1​​,*s*​*i*​​<*t*​*i*​​.

**Input Format**

The first line contains string 𝑠*s* (1≤∣𝑠∣≤100)(1≤∣*s*∣≤100) , consisting of lowercase English letters. Here, ∣𝑠∣∣*s*∣ denotes the length of the string.

The second line contains string 𝑡*t* (∣𝑡∣=∣𝑠∣)(∣*t*∣=∣*s*∣), consisting of lowercase English letters.

It is guaranteed that the lengths of strings s and t are the same and string 𝑠*s* is lexicographically less than string 𝑡*t*.

**Output Format**

If the string that meets the given requirements doesn't exist, print a single string "No such string" (without the quotes).

If such string exists, print it. If there are multiple valid strings, you may print any of them.

**Sample test**

|  |
| --- |
| **input** |
| K  m |
| **output** |
| l |

|  |
| --- |
| **input** |
| Klmnopq  klmpopq |
| **output** |
| klmnopr |

|  |
| --- |
| **input** |
| Abcde  abcdf |
| **output** |
| No such string |

1. **Arrays**

**Difficulty:**Secret   **Source:**Codeforces   **Time Limit:**2000ms   **Memory Limit:**256MB

You are given two arrays 𝐴*A* and 𝐵*B* consisting of integers, sorted in non-decreasing order. Check whether it is possible to choose 𝑘*k* numbers in array 𝐴*A* and choose 𝑚*m* numbers in array 𝐵*B* so that any number chosen in the first array is strictly less than any number chosen in the second array.

**Input Format**

The first line contains two integers 𝑛𝐴*n*​*A*​​, 𝑛𝐵*n*​*B*​​ (1≤𝑛𝐴,𝑛𝐵≤105)(1≤*n*​*A*​​,*n*​*B*​​≤10​5​​), separated by a space — the sizes of arrays 𝐴*A* and 𝐵*B*, correspondingly.

The second line contains two integers 𝑘*k* and 𝑚*m* (1≤𝑘≤𝑛𝐴,1≤𝑚≤𝑛𝑏)(1≤*k*≤*n*​*A*​​,1≤*m*≤*n*​*b*​​), separated by a space.

The third line contains 𝑛𝐴*n*​*A*​​ numbers 𝑎1,𝑎2,...,𝑎𝑛𝐴*a*​1​​,*a*​2​​,...,*a*​*n*​*A*​​​​ (−109≤𝑎1≤𝑎2≤...≤𝑎𝑛𝐴≤109)(−10​9​​≤*a*​1​​≤*a*​2​​≤...≤*a*​*n*​*A*​​​​≤10​9​​), separated by spaces — elements of array 𝐴*A*.

The fourth line contains 𝑛𝐵*n*​*B*​​ integers 𝑏1,𝑏2,...,𝑏𝑛𝐵*b*​1​​,*b*​2​​,...,*b*​*n*​*B*​​​​ (−109≤𝑏1≤𝑏2≤...≤𝑏𝑛𝐵≤109)(−10​9​​≤*b*​1​​≤*b*​2​​≤...≤*b*​*n*​*B*​​​​≤10​9​​), separated by spaces — elements of array 𝐵*B*.

**Output Format**

Print "YES" (without the quotes), if you can choose 𝑘*k* numbers in array 𝐴*A* and 𝑚*m* numbers in array 𝐵*B* so that any number chosen in array 𝐴*A* was strictly less than any number chosen in array 𝐵*B*. Otherwise, print "NO" (without the quotes).

**Sample test**

|  |
| --- |
| **Input** |
| 3 3  2 1  1 2 3  3 4 5 |
| **output** |
| YES |

|  |
| --- |
| **input** |
| 3 3  3 3  1 2 3  3 4 5 |
| **output** |
| NO |

|  |
| --- |
| **input** |
| 5 2  3 1  1 1 1 1 1  2 2 |
| **output** |
| YES |

**Explanation for sample test**

In the first sample test you can, for example, choose numbers 11 and 22 from array 𝐴*A* and number 33 from array 𝐵*B* (1<31<3 and 2<32<3).

In the second sample test the only way to choose 𝑘*k* elements in the first array and 𝑚*m* elements in the second one is to choose all numbers in both arrays, but then not all the numbers chosen in 𝐴*A* will be less than all the numbers chosen in 𝐵*B*.

1. **Big Segment**

**Difficulty:**Secret   **Source:**Codeforces   **Time Limit:**2000ms   **Memory Limit:**256MB

A coordinate line has 𝑛*n* segments, the 𝑖*i*-th segment starts at the position 𝑙𝑖*l*​*i*​​ and ends at the position 𝑟𝑖*r*​*i*​​. We will denote such a segment as [𝑙𝑖,𝑟𝑖][*l*​*i*​​,*r*​*i*​​].

You have suggested that one of the defined segments covers all others. In other words, there is such segment in the given set, which contains all other ones. Now you want to test your assumption. Find in the given set the segment which covers all other segments, and print its number. If such a segment doesn't exist, print −1−1.

Formally we will assume that segment [𝑎,𝑏][*a*,*b*] covers segment [𝑐,𝑑][*c*,*d*], if they meet this condition 𝑎≤𝑐≤𝑑≤𝑏*a*≤*c*≤*d*≤*b*

**Input Format**

The first line contains integer 𝑛*n* (1≤𝑛≤105)(1≤*n*≤10​5​​) — the number of segments. Next 𝑛*n* lines contain the descriptions of the segments. The 𝑖*i*-th line contains two space-separated integers 𝑙𝑖,𝑟𝑖*l*​*i*​​,*r*​*i*​​ (1≤𝑙𝑖≤𝑟𝑖≤109)(1≤*l*​*i*​​≤*r*​*i*​​≤10​9​​) — the borders of the 𝑖*i*-th segment.

It is guaranteed that no two segments coincide.

**Output Format**

Print a single integer — the number of the segment that covers all other segments in the set. If there's no solution, print −1−1.

The segments are numbered starting from 11 in the order in which they appear in the input.

**Sample test**

|  |
| --- |
| **input** |
| 3  3 3  4 4  5 5 |
| **output** |
| -1 |

|  |
| --- |
| **input** |
| 6  1 5  2 3  1 10  7 10  7 7  10 10 |
| **output** |
| 3 |

1. **Passwords**

**Difficulty:**Secret   **Source:**Codeforces   **Time Limit:**2000ms   **Memory Limit:**512MB

Vanya is managed to enter his favourite site Codehorses. Vanya uses 𝑛*n* distinct passwords for sites at all, however he can't remember which one exactly he specified during Codehorses registration.

Vanya will enter passwords in order of non-decreasing their lengths, and he will enter passwords of same length in arbitrary order. Just when Vanya will have entered the correct password, he is immediately authorized on the site. Vanya will not enter any password twice.

Entering any passwords takes one second for Vanya. But if Vanya will enter wrong password 𝑘*k* times, then he is able to make the next try only 55 seconds after that. Vanya makes each try immediately, that is, at each moment when Vanya is able to enter password, he is doing that.

Determine how many seconds will Vanya need to enter Codehorses in the best case for him (if he spends minimum possible number of second) and in the worst case (if he spends maximum possible amount of seconds).

**Input Format**

The first line of the input contains two integers 𝑛*n* and 𝑘*k* (1≤𝑛,𝑘≤100)(1≤*n*,*k*≤100) — the number of Vanya's passwords and the number of failed tries, after which the access to the site is blocked for 55 seconds.

The next 𝑛*n* lines contains passwords, one per line — pairwise distinct non-empty strings consisting of latin letters and digits. Each password length does not exceed 100100 characters.

The last line of the input contains the Vanya's Codehorses password. It is guaranteed that the Vanya's Codehorses password is equal to some of his 𝑛*n* passwords.

**Output Format**

Print two integers — time (in seconds), Vanya needs to be authorized to Codehorses in the best case for him and in the worst case respectively.

**Sample test**

|  |
| --- |
| **input** |
| 5 2  Cba  Abc  bb1  abC  ABC  abc |
| **output** |
| 1 15 |

|  |
| --- |
| **input** |
| 4 100  11  22  1  2  22 |
| **output** |
| 3 4 |

**Explanation for sample test**

* Consider the first sample case. As soon as all passwords have the same length, Vanya can enter the right password at the first try as well as at the last try. If he enters it at the first try, he spends exactly 11 second. Thus in the best case the answer is 11.
* If, at the other hand, he enters it at the last try, he enters another 44 passwords before. He spends 22 seconds to enter first 22 passwords, then he waits 55 seconds as soon as he made 22 wrong tries. Then he spends 22 more seconds to enter 22 wrong passwords, again waits 55 seconds and, finally, enters the correct password spending 11 more second. In summary in the worst case he is able to be authorized in 1515 seconds.
* Consider the second sample case. There is no way of entering passwords and get the access to the site blocked. As soon as the required password has length of 22, Vanya enters all passwords of length 11 anyway, spending 22 seconds for that. Then, in the best case, he immediately enters the correct password and the answer for the best case is 33, but in the worst case he enters wrong password of length 22 and only then the right one, spending 44 seconds at all.

1. **Suffix Structures**

**Difficulty:**Secret   **Source:**Codeforces   **Time Limit:**1000ms   **Memory Limit:**512MB

Bizon the Champion isn't just a bison. He also is a favorite of the "Bizons" team.

At a competition the "Bizons" got the following problem: "You are given two distinct words (strings of English letters), 𝑠*s* and 𝑡*t*. You need to transform word 𝑠*s* into word 𝑡*t*".

The task looked simple to the guys because they know the suffix data structures well. Bizon Senior loves *suffix automaton*. By applying it once to a string, he can **remove** from this string any single character. Bizon Middle knows *suffix array* well. By applying it once to a string, he can **swap** any two characters of this string. The guys do not know anything about the suffix tree, but it can help them do much more.

Bizon the Champion wonders whether the "Bizons" can solve the problem. Perhaps, the solution do not require both data structures. Find out whether the guys can solve the problem and if they can, how do they do it? Can they solve it either only with use of *suffix automaton* or only with use of *suffix array* or they need both structures? Note that any structure may be used an unlimited number of times, the structures may be used in any order.

**Input Format**

* The first line contains a non-empty word 𝑠*s*. The second line contains a non-empty word 𝑡*t*.
* Words 𝑠*s* and 𝑡*t* are different. Each word consists only of lowercase English letters. Each word contains at most 100100 letters.

**Output Format**

* In the single line print the answer to the problem:
  + Print "need tree" (without the quotes) if word 𝑠*s* cannot be transformed into word even with use of both *suffix array* and *suffix automaton*.
  + Print "automaton" (without the quotes) if you need only the *suffix automaton* to solve the problem.
  + Print "array" (without the quotes) if you need only the *suffix array* to solve the problem.
  + Print "both" (without the quotes), if you need both data structures to solve the problem.
* It's guaranteed that if you can solve the problem only with use of *suffix array*, then it is impossible to solve it only with use of *suffix automaton*. This is also true for *suffix automaton*.

**Sample test**

|  |
| --- |
| **input** |
| automaton  tomat |
| **output** |
| automaton |

|  |
| --- |
| **input** |
| array  arary |
| **output** |
| array |

|  |
| --- |
| **input** |
| Both  hot |
| **output** |
| both |

|  |
| --- |
| **input** |
| Need  tree |
| **output** |
| need tree |

**Explanation for sample test**

In the third sample you can act like that: first transform "both" into "oth" by removing the first character using the *suffix automaton* and then make two swaps of the string using the *suffix array* and get "hot".