## Sequences of Equal Strings

Write a program that reads an array of strings and finds in it all sequences of equal elements (comparison should be **case-sensitive**). The input strings are given as a single line, separated by a space. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| hi yes yes yes bye | hi  yes yes yes  bye |
| SoftUni softUni softuni | SoftUni  softUni  softuni |
| 1 1 2 2 3 3 4 4 5 5 | 1 1  2 2  3 3  4 4  5 5 |
| a b b xxx c c c | a  b b  xxx  c c c |
| hi hi hi hi hi | hi hi hi hi hi |
| hello | hello |

## Longest Increasing Sequence

Write a program to find all **increasing** sequences inside an array of integers. The integers are given on a single line, separated by a space. Print the sequences in the order of their appearance in the input array, each at a single line. Separate the sequence elements by a space. Find also the longest increasing sequence and print it at the last line. If several sequences have the same longest length, print the **left-most** of them. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 3 4 1 50 **2 3 4 5** | 2 3 4  1 50  2 3 4 5  Longest: 2 3 4 5 |
| **8 9** 9 9 -1 5 2 3 | 8 9  9  9  -1 5  2 3  Longest: 8 9 |
| **1 2 3 4 5 6 7 8 9** | 1 2 3 4 5 6 7 8 9  Longest: 1 2 3 4 5 6 7 8 9 |
| 5 **-1 10 20** 3 4 | 5  -1 10 20  3 4  Longest: -1 10 20 |
| **10** 9 8 7 6 5 4 3 2 1 | 10  9  8  7  6  5  4  3  2  1  Longest: 10 |

## Collect the Coins

Working with multidimensional arrays can be (and should be) fun. Let's make a game out of it.

You receive the layout of a **board** from the console. Assume it will always have **4 rows** which you'll get as strings, each on a separate line. Each character in the strings will represent a **cell** on the board. Note that the strings may be of different length.

You are the player and start at the top-left corner (that would be position **[0, 0]** on the board). On the fifth line of input you'll receive a string with movement commands which tell you where to go next, it will contain only these four characters – '**>**' (move right), '**<**' (move left), '**^**' (move up) and '**v**' (move down).

You need to keep track of two types of events – collecting coins (represented by the symbol '**$**', of course) and hitting the walls of the board (when the player tries to move off the board to invalid coordinates). When all moves are over, **print the amount of money** collected and the **number of walls hit**. Example:

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| Sj0u$hbc  $87yihc87  Ewg3444  $4$$  V>>^^>>>VVV<< | Coins collected: 2  Walls hit: 2 | Starting from (0, 0), move down (coin), twice right, up, up again (wall), three times right (coin on second move), twice down, down again (wall), twice to the left – game over (no more moves). Total of two coins collected and two walls hit in the process. |

## Phonebook

Write a program that receives some info from the console about **people** and their **phone numbers**.

You are free to choose the manner in which the data is entered; each **entry** should have just **one name** and **one number** (both of them strings).

After filling this simple phonebook, upon receiving the **command** "**search**", your program should be able to perform a search of a contact by name and print her details in format "**{name} -> {number}**". In case the contact isn't found, print "**Contact {name} does not exist.**" Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| Nakov-0888080808  **search**  Mariika  Nakov | Contact Mariika does not exist.  Nakov -> 0888080808 |
| Nakov-+359888001122  RoYaL(Ivan)-666  Gero-5559393  Simo-02/987665544  **search**  Simo  simo  RoYaL  RoYaL(Ivan) | Simo -> 02/987665544  Contact simo does not exist.  Contact RoYaL does not exist.  RoYaL(Ivan) -> 666 |

\* **Bonus:** What happens if the user enters the same name twice in the phonebook? Modify your program to keep **multiple phone** numbers per contact.