2082 Roman Forts

This program will acquaint you with the judicious method used by rulers of the ancient Roman empire to choose the locations of their military forts.

You are given a collection of cities.

- The cities will be labeled by upper case letters in alphabetical order, starting with 'A'.
- Given any pair of cities, there is one and only one path between them (see Figure 1).
  - This path may be a direct connection (as between J and C in Figure 1) or it may pass through a number of intermediate cities (as between J and B in Figure 1).
  - In the example of Figure 1, the path from J to B passes through C, G, and F.

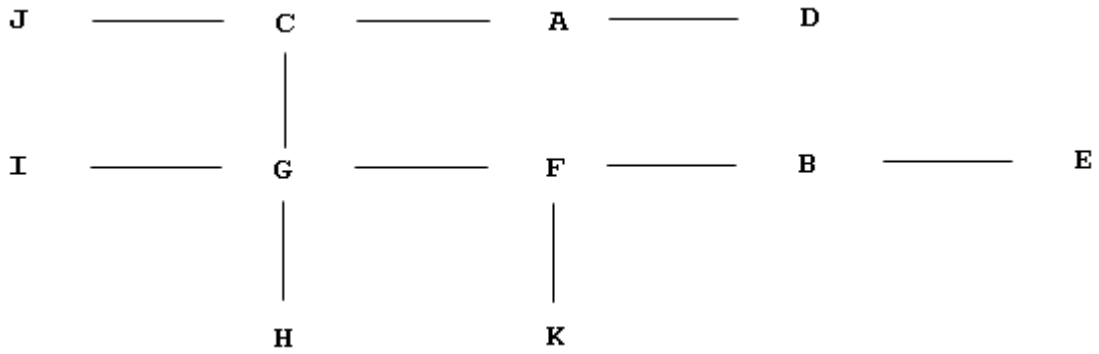

Figure 1

  - The actual number of miles between cities is irrelevant. The *distance* between a pair of cities is defined as the number of hops (each "hop" being taken over a over a direct connection) between them.
    - In Figure 1, the number of hops between J and C is 1, and the number of hops between J and B is 4.
  - Initially, only one of the cities is *fortified* (by a military fort). As your program executes, additional cities will be fortified. v If a city is not fortified, we say it is *vulnerable*.
    - Furthermore, if a city is vulnerable, its *degree of vulnerability* is defined as the shortest distance between it and a fortified city.
  - When one or more cities have already been fortified, the location of an additional fort is chosen by a simple rule: it is the most vulnerable city (the one having the highest degree of vulnerability).
    - In case of a tie, choose the alphabetically first city among those tied as the most vulnerable city.

Input

The first line of the input file contains three items of information, separated by one or more blank spaces:

- The total number of cities in the given collection. It will be in the range 3...26. In this example, it is 11, which means that the cities are labeled '`A`' ... '`K`'.

- An upper case letter in the valid range of cities, denoting the city which is initially fortified.
- The total number of cities that are to be fortified, including the initially fortified one. It will be at least three and not greater than the total number of cities in the collection.

Each of the remaining lines of the input file will contain two upper case letters, separated by one or more blank spaces.

- Each letter will be in the range of labels of cities in the given collection.
- Each pair of letters on one line of input determines a direct connection between two cities.
  - The direct connections listed in the input file shown on the right is identical to the direct connections of Figure 1.
- The set of direct connections listed in the input file will satisfy all previously stated conditions.

All lines of input will be free of leading or trailing blank spaces.

Output

The output, written by your program, will list the cities to be fortified, as upper case letters, in the correct order.

Explanation of the sample output:

- When there is only one fortified city, the degree of vulnerability of each vulnerable city is its distance from the fortified city.
  - Initially, with '`J`' being the only fortified city, '`E`' is the most vulnerable city with degree of vulnerability 5.
- With '`J`' and '`E`' being fortified, '`D`', '`H`', '`I`', and '`K`' contend for being the most vulnerable city, all having degree of vulnerability 3.
  - For example, the distance from '`D`' to '`E`' is 6, the distance from '`D`' to '`J`' is 3; the smaller of these two numbers is 3. Therefore, the degree of vulnerability of D is 3.
  - The alphabetically first city among the four contenders ('`D`', '`H`', '`I`', and '`K`') is '`D`', which will be the next city to be fortified.
- Similarly, '`H`', and then '`K`', will be selected as the next city to be fortified. At this point, the number of fortified cities is 5 (as specified in the input), and the program terminates
  - Note that city I is not fortified, because its degree of vulnerability drops from 3 to 2 once '`H`' is fortified, making '`K`' the most vulnerable city.
- Formatting details:
  - The upper case letters denoting the cities in the output will be separated by one blank space.
  - There will not be any blank lines or leading blank spaces in the output.

Sample Input

```
11 J 5
A D
B E
F B
C G
F G
C A
G H
G I
J C
K F
```

Sample Output

```
J E D H K
```