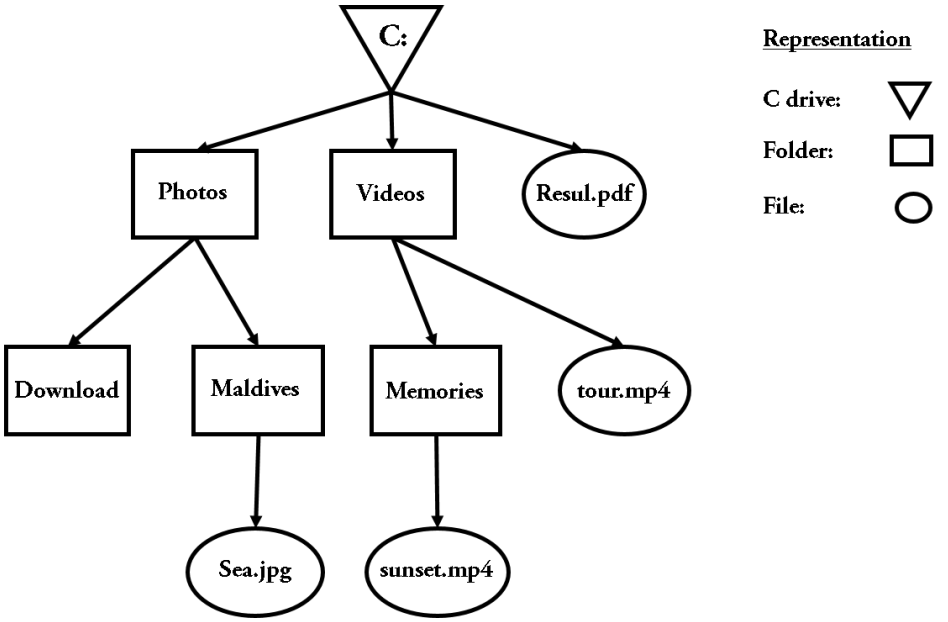
**Military Institute of Science and Technology**
**Department of Computer Science and Engineering**
**Level-2, Spring Term, 2021**
**CSE-204 (Data Structures and Algorithms 1)**
**Practice Problems**

## Problem 1

A file system in the Windows operating system generally consists of a number of drives like C:, D:, E: …….. etc. But consider a file system that consists of only "**C:**" drive. All the folders and files are located inside the "**C:**" drive.

You know that a file system inside a computer always follows a tree data structure. See the following figure to understand a file system properly.




Now a file system is given to you as input. You have to determine the following:

1. Absolute path of a given folder/file starting from the C: drive
2. Number of mouse clicks required to access a given file/folder starting from C: drive [You can assume that 2 mouse clicks are required to access a folder/file]

For example the absolute path of "Sea.jpg" from the figure is: C:\\Photos\\Maldives\\Sea.jpg and a total of 6 mouse clicks will be required to access the file starting from C drive.

Consider another example, the absolute path of "Memories" from the figure is: C:\\Videos\\Memories and a total of 4 mouse clicks will be required to access the folder.

## Input

First line of input contains a single integer ***e*** denoting the number of relationships in the file system. Next ***e*** lines contain 2 strings ***X*** and ***Y*** each separated by a space denoting that ***Y*** is a folder/file inside ***X***. Next line contains an integer ***q*** denoting the number of queries. Next ***q*** lines contain a string ***A***. For simplicity you can assume that the name of the folders/files are unique and contain only letters and dot (.) and all the files/folders have unique names. A string "C:" as ***X*** represents the C: drive.

## Output

For each query print the absolute path of ***A*** in the first line and then print the number of required mouse clicks to access ***A*** starting from C: drive. If ***A*** is not found in the given file system then print "File not found" and print "-1" in the next line. See the Sample Input Output for clarification.

**Sample Input Output**

| Input | Output |
|---|---|
| 9<br>Photos Maldives<br>C: Photos<br>Videos Memories<br>C: Videos<br>Memories sunset.mp4<br>Maldives Sea.jpg<br>Videos tour.mp4<br>C: Result.pdf<br>Photos Download<br>5<br>Sea.jpg<br>Memories<br>tour.mp4<br>Photos<br>Codes | C:\\Photos\\Maldives\\Sea.jpg<br>6<br><br>C:\\Videos\\Memories<br>4<br><br>C:\\Videos\\tour.mp4<br>4<br><br>C:\\Photos<br>2<br><br>File not found<br>-1 |

## Problem 2

A Graph is called **Bicolorable** or **Bipartite** if all the vertices of the graph can be colored with exactly 2 different colors where no two adjacent vertices have the same color. Consider the following figures:

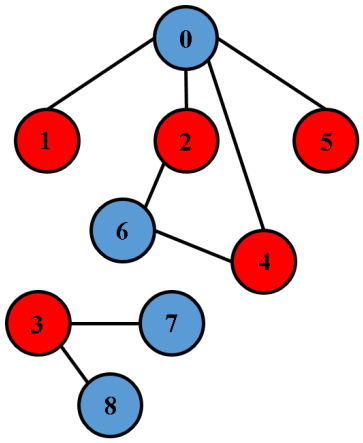



**Figure-1**

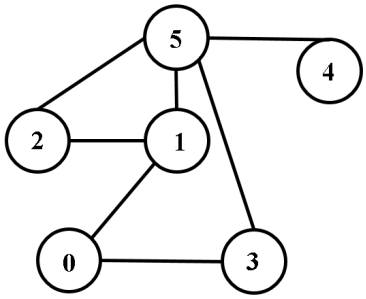



**Figure-2**

The graph of Figure-1 is Bicolorable because all the vertices are colored with either Red or Blue and no two adjacent vertices are colored with the same color. But if you consider the graph in Figure-2, that is not Bicolorable because Vertex-1, Vertex-2 and Vertex-5 can't be colored with 2 different colors where all the adjacent vertices have different colors. Now the task is to find whether a given undirected graph is bicolorable or not.

### Input

The first line of input contains 2 integers ***n*** and ***e***. **n** denotes the number of vertices and ***e*** denotes the number of edges. Next ***e*** lines of input contain 2 integers ***X*** and ***Y*** each denoting that there is an edge between ***X*** and ***Y.*** You can safely assume that ***X*** and ***Y*** lie between 0 to ***n***-1.

### Output

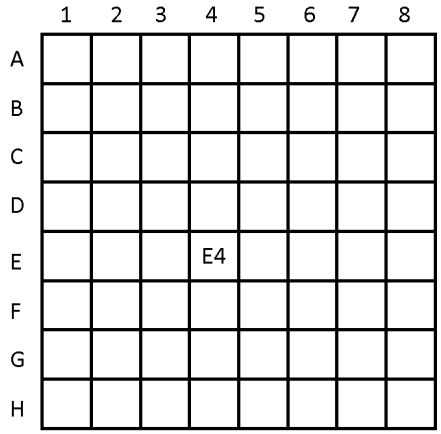
Print "YES" if the given graph is Bicolorable otherwise print "NO".

### Sample Input Output

| Input | Output |
|---|---|
| 9 8<br>0 1<br>2 0<br>6 2<br>3 7<br>8 3<br>4 6<br>4 0<br>0 5 | YES |
| 6 7<br>1 2<br>5 4<br>5 1<br>2 5<br>5 3<br>1 0<br>0 3 | NO |

## Problem 3

A knight is allowed to move 2.5 cells on a chessboard. A chessboard is a 8x8 board where the rows are marked from A to H and columns are marked from 1 to 8. A cell is read by its row letter followed by its column number. Following figure is a demonstration of a chess board and a cell E4 is marked for your convenience so that you can understand the cell naming convention.



Now consider that a Knight is currently located on E4. Then at the next step the Knight can move at any of the following cells: F2, D2, F6, D6, C3, C5, G3, G5. Now your task is very simple. You are given the initial position of a Knight on the chessboard. You have to determine the minimum number of required moves for the Knight to move from its initial position to a given destination along with its path. The solution can have multiple path, in that case you print any one of them
[Similar online judge problem: UVA 439]

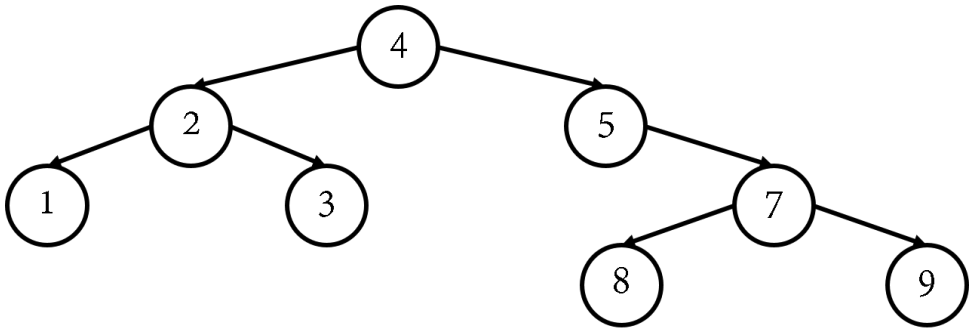
### Input
First line contains a single integer ***q*** denoting the number of queries. Next ***q*** lines contain 2 space separated strings ***X*** and ***Y***. ***X*** represents the initial cell and ***Y*** represents the destination cell of the Knight.

### Output
For each query print the minimum number of moves required for the Knight to move from cell-***X*** to cell-***Y*** and in the next line print any of the possible paths.

### Sample Input Output

| Input | Output |
|---|---|
| 2<br>A1 B2<br>E2 E4 | 4<br>A1->B3->C1->D3->B2<br><br>2<br>E2->C3->E4 |

## Problem 4
It's not an easy task to print a Binary Search Tree on the computer. But an easy way to visualize a Binary Search Tree is to print the values level wise from left to right. Consider the following Binary Search Tree considering 4 as root.



The level wise (left to right) traversal of the above BST is as the following:
4
2 5
1 3 7
8 9
Yes! This traversal never can express the parent-child relationship of a Binary Search Tree properly but a simple visualization can be done by this traversal. Now to implement this traversal first you will have to know the method of inserting values in a Binary Search Tree and then you will have to apply a breadth first search on the tree from left to right. Printing a newline after printing all the values of a level is very essential for this representation. So you must keep this in mind.

### Input
First line of input contains a single value ***n*** denoting the number of nodes/values in the BST. Then the next line of input contains total ***n*** space separated integers that need to be inserted sequentially in a BST.

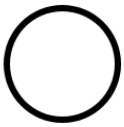
### Output
Print the level wise traversal of the BST from left to right starting from the root. You must print a newline after printing all the values in a level.

### Sample Input Output

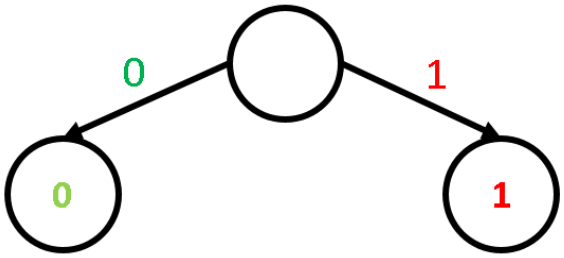
| Input | Output |
|---|---|
| 8<br>4 5 2 7 1 3 8 9 | 4<br>2 5<br>1 3 7<br>8 9 |

## Problem 5
A BFS algorithm is widely used to generate all possible combinations of a problem. For example: generating all the bit strings of length ***n***. Now step by step simulation for generating all the bit strings of length 3 is shown below.

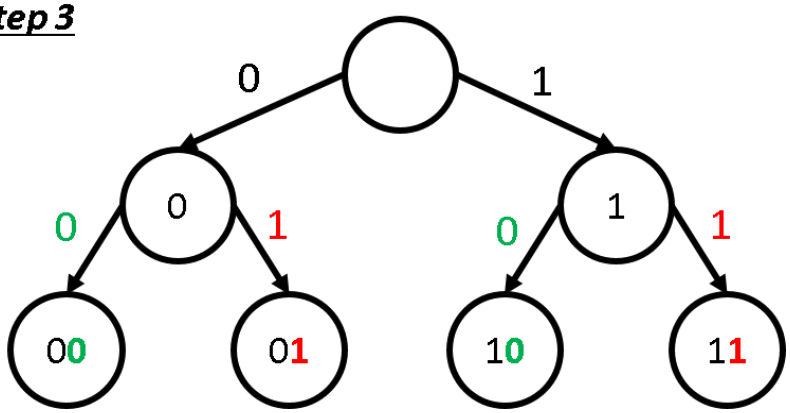
***Step 1***



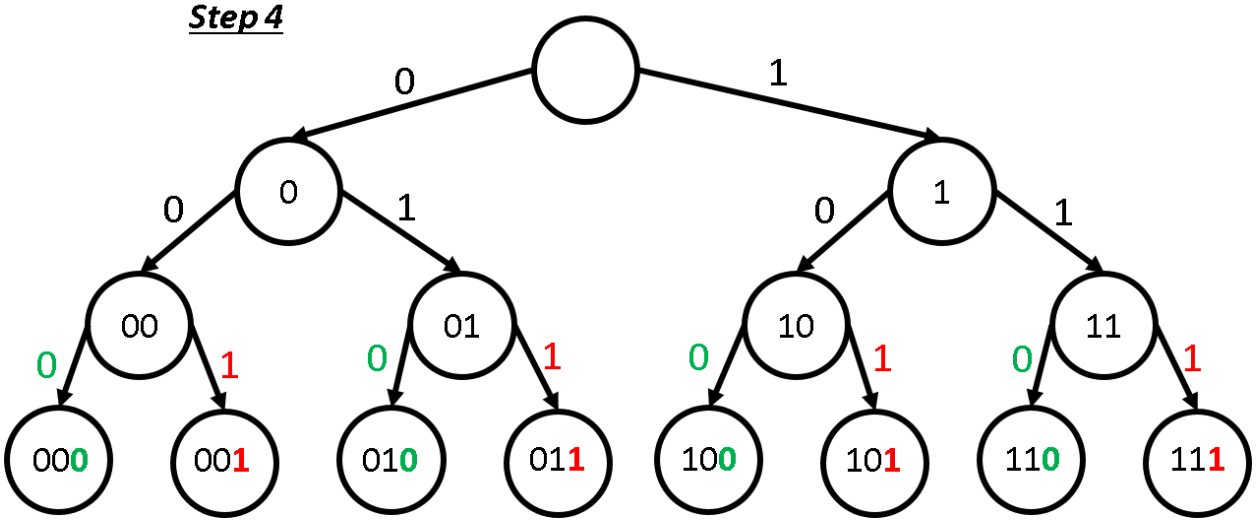
***Step 2***




***Step 3***




***Step 4***



A “0” is concatenated at the right to form the left child and a “1” is concatenated at the right to form the right child. Now you are given an integer ***n*** and your task is to print all the binary strings of length 1, length 2, length 3 …… length ***n***.
[Don’t execute the BFS algorithm multiple times. Execute for single time and do your required task]

### Input
Only line of input contains a single integer ***n***.

### Output
Print all the bit strings of length ***i*** where ***i*** = 1 to ***n***. See the Sample Input Output for clarification.

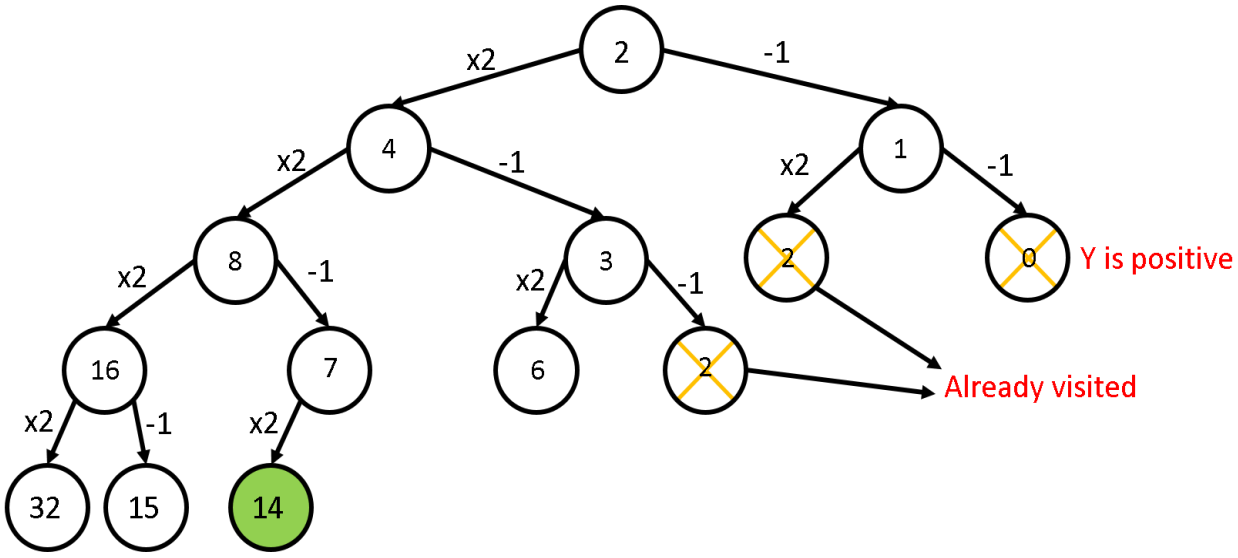
### Sample Input Output

| Input | Output |
|---|---|
| 3 | Length 1: 0 1<br>Length 2: 00 01 10 11<br>Length 3: 000 001 010 011 100 101 110 111 |
| 4 | Length 1: 0 1<br>Length 2: 00 01 10 11<br>Length 3: 000 001 010 011 100 101 110 111<br>Length 4: 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 |

## Problem 6
A BFS algorithm is widely used to generate all possible combinations of a problem. Let 2 positive integers ***X*** and ***Y*** are given to you. You are allowed to do 2 operations at each step on a number ***P***

1. ***P*** = ***P*** *2
2. ***P*** = ***P***-1

Now your task is to calculate the minimum number of operations required to convert ***X*** into ***Y*** along with its conversion path. A simulation is shown in the following figure considering ***X***=2 and ***Y***=14.




Here, minimum required steps = 4
and the conversion path is: 2 -> 4 -> 8 -> 7 -> 14
[Similar online judge problem: Codeforces-520B]

## Input

Only line of input contains 2 space separated positive integers ***X*** and ***Y***.

## Output

Print the minimum number of steps required to convert ***X*** into ***Y*** and then print its conversion path in the next line. See the sample Input Output section to understand the output format.

## Sample Input Output

| Input | Output |
|---|---|
| 2 14 | 4<br>2 -> 4 -> 8 -> 7 -> 14 |
| 4 14 | 3<br>4 -> 8 -> 7 -> 14 |