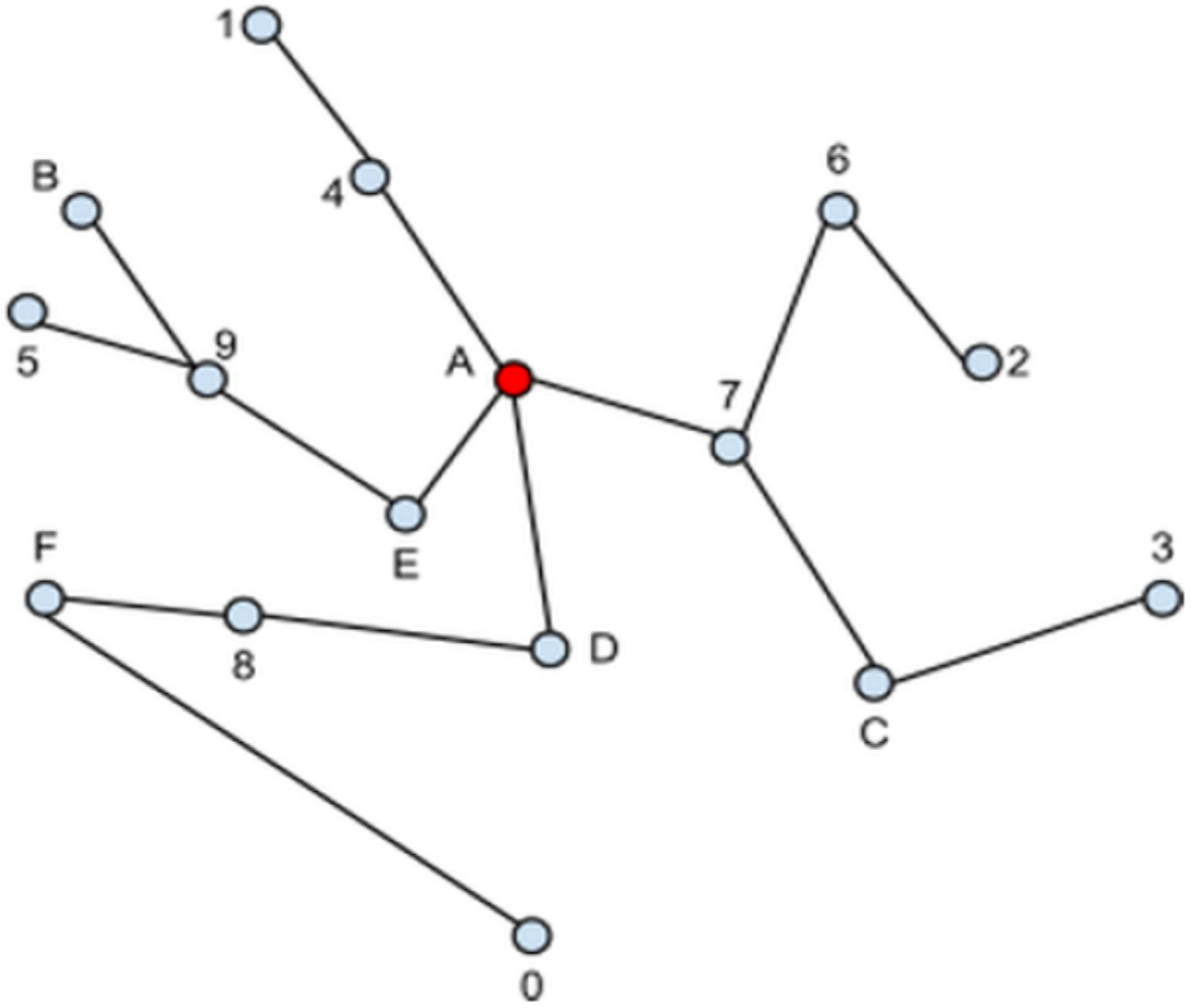
Chapter-1

Intelligence tells us that Doctor Ja'koof is somewhere in the US. In order to track him down, we need an optimized transit plan for our field operatives. This map shows **16** major cities in US and their connectivity.



Field Agent Kim is in city A, and to collect information he has to visit all of the cities on the map. In what order should he visit them so that the total travel time is minimized (he doesn't need to come back to A)? It takes one unit of time to go from a city to another city directly connected to it. If more than one order exists, he prefers the lexicographic smallest one. Letters are lexicographically smaller than digits (A-Z comes before 0-9).

If Kim visits the city in the following order: A-7-C-3-C-7-6-2-6-7-A-4-1-4-A-D-8-F-0-F-8-D-A-E-9-B-9-5
the output will be: **A7C36241D8F0E9B5**

The output shows the order in which a new city has been visited.

Task
What's the optimal order to visit the cities? Paste the output string below.

==========

Chapter 2

The sweep of the cities did not lead to the capture of the Doctor. Although the mission did not meet its objectives, it was not a
complete failure. We have located the Doctor's hidden warehouse. When we got there most of the evidence was destroyed, but we were able to recover a few drives.

Next to the computer was a post-it note that read
"Huh, and you thought by chasing me from city to city you could catch me? Even my shadow fears to follow me! The password to my computer is a bijection of the cities you visited. Crack it if you can… Bye!"

The bijection function (f:D->R) is a mapping from set {A, B, C,.. F, 0, 1, 2,.., 9} to set {A, B, .., F, 0, 1, 2,…, 9}. The bijection **f** set by the Doctor is the smallest possible lexicographic string such that

$$f^k(S) = S : \ for\ some\ k > 50$$

and

$$f^k(S) \neq S \ \forall \ 1 < k \leq 50$$

where **S** is the order of cities you visited and **k** indicates a composite function. Can you figure out the bijection mapping used by the Doctor following this criteria? Letters have a smaller lexicographical value than numbers.
Two sample f are :
ABCDEF0123456789 => DE05A97624B1C38F
ABCDEF0123456789 => D90BAE36245C178F
and DE05A97624B1C38F is lexicographic smaller than D90BAE36245C178F
Examples
If the bijection function **f** is: ABCDEF0123456789 => DE05A97624B1C38F

and let's say the order of cities visited (S) is: ABCD56789EF01234
f(S) = DE051C38FA97624B
$f_2(S)$ = 5A7160489DF3C2BE
and so on..

Task

Print the image of f(S) where S is the order of cities you visited

Example output

DE05A97624B1C38F

==========

Chapter 3

You finally cracked the code to enter the Doctors computer and snoop around! You open every possible file and browse his logs. You even search his web history until IE craps out on you.

No luck… Seems like there is nothing of importance on the computer. There are other things you investigate in the room - breaking open a briefcase, opening up all the drawers, searching the attic. In the basement there is something that stands out… A vault like door with a digital lock on it. It looks like it's impossible to open with out the code.

On looking into Doctor's computer, you have a hunch that the file might contain the password for the metal door. But the only way to read the data in the file is by writing a program that takes this file as an input.

On debugging the output of your program you realise that the file has only hashes and dots, like a Captcha. It seems that all of the characters are a part of the CipherKey.
Wow! Can this be the key to the door?

Task

You are given a text input consisting of dots and hashes which contains a subset of the characters of CipherKey and not necessarily in the same order. The function **getDoorPass** takes in the array of strings consisting of dots & hashes, the length of each string, and the number of strings as input. Complete the function to return the string found in the array which will be the key to open the locked door.

Input Format

The first line contains
x y => The row and column size

Followed by x rows of y characters.
Your program takes the captcha like image as input

```
8 52
....................................................
..#...####.....................................####....
..#.....#..#####..#..#.....#####.........#.......
..#...####.....#..#..#.....#.............#.......
..#...#.....#####..####.....#####.........####....
..#...####.....#.....#.........#.........#..#....
..#.........#####.....#.....#####.........####....
....................................................
```

And the **Output** must be

```
123456
```

*Note:-The numbers represented in the sample testcase is for reference only.

==========

Chapter 4

Congratulations! The lock opened! You open the door to find a room filled with pile after pile of papers stacked waist high. Each of the sheets contain just three words - “BRAIN”, “F***” and “DANGER”. The back of each paper contains a random code snippet. The snippet is present in the editor below. Yes, it’s a programming language!

Task
This code uses the last 3 characters from the password that opened the door as input. You’ve two things todo:
a. *Compile & Test* the code below by specifying the input in the *custom testcase* textbox. The output will tell you to fix a bug.
b. Fix the bug and rerun it with the input. Maybe we’ll find out what the “DANGER” means…

==========

Chapter 5

While trying to figure out the BrainFuck code, a stack of papers falls over.

!!!!! BOOM !!!!!

The floor is rigged to explode! Headquarters sends in a specialized robot that can sense the presence of the explosives in the floor, and neutralize them by placing boxes over the bombs and containg the explosions. The bot can move left (l), right(r), front (u), back (d). Program the bot to navigate the room and place boxes over the bombs. When the bot is adjacent to a box, it can move it by pushing it in direction of the robot and the box. If it hits a wall, the box cannot be moved any farther in that direction.

INPUT FORMAT

```
11 19
<map of the room with the input parameters>
```

First line is the size of the input. 11 lines follow each with 19 characters. Each character represents a square tile which can be one of the four things that you get as an output from the BrainFucked challenge- wall or box or bomb or initial position of the robot. The robot moves one step at a time.

OUTPUT FORMAT

```
(lrud)*
```

The output is a series of *lrud* moves the robot makes from the initial state to the final state. The moves must end when all the boxes are placed over the position where the bombs are placed.

Task

Complete the function *moves* that takes the map as the input and returns a string of moves made by the robot as the output.

==========

Chapter 6

Domineering is a mathematical game played between two players who alternate by placing a domino of size 2x1 on an square board. The first player places the dominoes vertically and the second player places the dominoes horizontally on the board. The player unable to place his domino on the board loses.

Input Format

The game takes place on an **8x8** board. The first player is represented by **v** (ascii value: 118) and the second player by **h** (ascii value: 104). Empty cell is denoted by - (ascii value: 45) and a domino occupied cell is denoted by either **v** or **h**

The top left of the board is (0,0) and the bottom right is (7,7)

The first line denotes the character of the player who is playing. The next 8 lines contains 8 characters each representing the board.

Sample Input

```
v
-v------
-vhh----
---v----
---v----
---hh---
--------
--------
--------
```

Sample Output

```
2 1
```

For the first player, a move is denoted by its upper square. Here, the first player to place the domino at (2,1) and (3,1) outputs the position as (2,1) which is its upper square.

Sample Input

```
h
-v------
-vhh----
-v-v----
-v-v----
---hh---
--------
--------
--------
```

Sample Output

```
4 1
```

For the second player, a move is denoted by its left square. Here, the second player to place the domino at (4,1) and (4,2) outputs the position as (4,1) which is its left square.

Complete the function **nextMove** that takes a *char* player and an 8x8 board as input to print the next move.

Your output must be two integers with a single space separating them.

==========

Chapter 7

“Consider yourself lucky to reach this stage. Many have tried to fight me, very few have come this far but none have won. Catch me if you can!!”

Dr. Jakoof is on the run! 3 Cops are chasing him on a rectangular grid like road network. Cops can only move 1 step in any of the 4 directions (north, south, east or west), but Dr. Jakoof can move diagonally as well. Catch Dr. Jakoof before he escapes.
But wait, there's a rat in the house who's helping Dr Jakoof plan his moves. Hmm.. so, the bot should be able to play as Dr Jakoof and the cop. As a cop, you must catch Dr. Jakoof within 150 moves. As Dr Jakoof, you must evade capture for utmost 150 moves.
The chase happens on a 20x20 grid. All the 3 cops start at (0,0) and Dr Jakoof starts at (19,19). The top left of the grid is (0,0) and the bottom right is (19,19).

Input format
The first line is a character. Cor or Dr. Jakoof. Cop is represented by the character 'C'. Dr. Jakoof is represented by the character 'R'.The next line contains 6 integers. The first two integers are the positions of Dr Jakoof, and the rest 6 are the positions of the 3 cops.
The moves are simultaneous. Cops move first and Dr Jakoof moves next and this repeats.
The positions given to the cops and Robber are the previous moves made by the cop and the robber. More information on how the simultaneous moves are handled are given atIO format.

Output format
If you are a Cop, output is 6 single spaced integers which are the next positions of the Cops 1, 2 and 3. If you are Dr Jakoof, the output is 2 single spaced integers which are the next position of Dr Jakoof.

Cops
Initial State
Sample Input
```
C
19 19 0 0 0 0 0 0
```

Sample Output
```
0 1 0 1 1 0
```

The first cop moves to (0,1) and the second cop moves to (0,1) and the third cop moves to (1,0) respectively. More than one cop can occupy the same cell.
Intermediate State
Sample Input
```
C
12 13 5 6 7 8 9 9
```

Sample Output

5 6 7 9 9 10

the first cop stays in his original position and not make a move. This is a valid output. The other two cops move from (7,8) to (7,9) and (9,9) to (9,10) respectively.

Dr Jakoof

Initial State

Sample Input

R
19 19 0 0 0 0 0 0

Sample Output

18 18

As you can see, Dr Jakoof is allowed to move diagonally as well. Dr Jakoof moves from (19,19) to (18,18).

Intermediate State

Sample Input

R
12 13 3 4 5 6 7 8

Sample Output

12 12

Dr Jakoof here makes a move from (12,13) to (12,12)

Complete the function **next_move** that takes in a char *player*, the positions of Dr Jakoof and the three positions of the cop and return the positions of the cop or Dr Jakoof respectively.

The bot is allowed to share data between moves by writing a file in the current directory and reading it in subsequent moves.