Round 1

This section consists of 4 problems.
You have 1 hour and 45 minutes to work on these problems.
Your output must EXACTLY match the output given in the samples.

The Lazy Time Traveler

True laziness is achieved when one spends more effort avoiding work than the work would have taken in the first place.

Description

A very lazy time traveler lives on the equator of the Earth. Like many time travelers, he loves to travel! Paris, Cairo, Disney World — it would be a waste to only explore different times! Unfortunately, our Mr. Time Traveler suffers from crippling laziness. Lucky for him, his time machine can help! If he simply travels in time, and lets the Earth rotate underneath his time machine, he can force places to come to him — brilliant!

Your task is simple — given a starting location (in degrees), and a desired ending location (again, in degrees), calculate how long the time traveler will need to travel in time to arrive at his destination. Remember, the Earth spins 360° every 24 hours.

Example Input/Output

You will be given a starting location $s \in [0, 360]$ and an ending location $e \in [0, 360]$, separated by spaces. You must output a time, in hours between -12 and 12.

```
input: 0 360
output: 0.0

input: 320 200
output: -8.0
```

Chutes and SPACE-LADDERS

While exploring the future, Seymore Time Traveler discovered a new technology — the LASER-AIDED-DIMENSIONAL-DRIFT-ENERGY-RIDER!

Description

Our hero is, yet again, in a predicament of laziness. He has discovered many exciting new worlds, connected together by *SPACE WORMHOLES*! He would love to travel through them. . . but he doesn't want to travel through *too* many wormholes (because that sounds hard).

Your task, should you choose to accept it, is to find the shortest path from the starting world to the destination of Seymore's choice.

Example Input/Output

Each of the first n lines of input will be a world name (a single upper-case character), followed by a list of each connected world. . The $n+1$'th line will be a single upper-case letter, designating the desired destination. You must output a sequence of letters that represents the shortest path to the destination. Assume you start at the first given world. You can assume a path will always exist. There will be less than 10 worlds given. If multiple valid paths exist, any will be accepted.

```
input:
A B D
B C D
C
output:
A B C
```

```
input:
A B D
B C D
A
output:
A
```

```
input:
A B C D
B F
D B C
F E
E
output:
A B C E
```