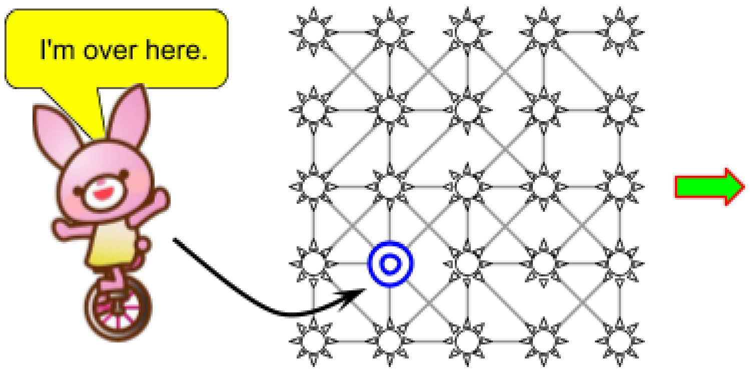
A Turn for the Worse

The Idea

Write a program in the language of your choice that controls a unicycler on a course to ride "edges" through a grid of partially connected locations travelling as far as possible, turning as few times as possible, and never riding the same edge twice. The input file will describe **six** grids of locations, edges, and the starting locations of the unicycler.

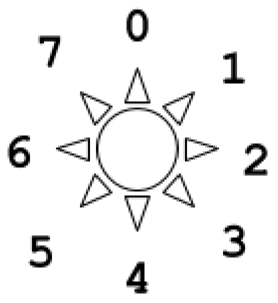
Below is an example of one grid of locations/edges and the equivalent portion of the input file your program would receive for it.



```
5
*-* * *-*
 X|X|X|\
*-*-* *-*
|/|/|X| |
*-*-*-*-*
|X|X|X X|
*-@ *-*-*
|\|X|\|X|
*-*-*-* *
```

The unicycler is controlled by only eight operational commands, each a single character:

- **0** : move one location to the North
- **1** : move one location to the Northeast
- **2** : move one location to the East
- **3** : move one location to the Southeast
- **4** : move one location to the South
- **5** : move one location to the Southwest
- **6** : move one location to the West
- **7** : move one location to the Northwest



The Goal

With the unicycler starting at the position dictated by the input with an '@' character, create a string of directional digits (listed above) to move the unicycler across edges to neighboring locations. Once you've crossed a specific edge between two locations, you cannot travel that edge again. You cannot travel from one location to a neighbor if there isn't an edge connecting them in the desired direction. Each time you successfully move from one location to another,

one point will be added to your score. If you changed directions from your previous direction in order to reach the next location, **one point** will be subtracted from your score, effectively gaining and losing nothing by having turned. Your goal is to earn the highest possible score you can for each of the **six** courses. Your final score for the challenge will be the sum of the **six** individual course scores, where each course's score is…

(# of valid moves from location to location) **-** (# of times you turned)

Input

An input file named "courses.txt" is provided to you during your development. This will be the exact same file that the judges will use to test your solutions as you submit them. The following describes the makeup of the input file...

- There will be **six** courses described, one after another, with each constructed the same way
    - The first line of a course description will be an integer identifying how many rows and columns of locations there are on the course. There is only one number as all courses are square, having the same numbers of rows and columns.
    - The next several (to many) lines will be the ASCII art representation of the course. Each row will be the same length, sometimes beginning and/or ending with space characters when appropriate to the map. It is in your best interest to examine the actual courses.txt file to really understand the format.
    - The characters you'll encounter in the ASCII art mean the following:
        - `'*'` : a **location** the unicycler might travel to
        - `'@'` : the initial **location** of the unicycler
        - `'-'` : an **edge** that can be travelled in the East or West directions
        - `'/'` : an **edge** that can be travelled in the Northeast or Southwest directions
        - `'|'` : an **edge** that can be travelled in the North or South directions
        - `'\'` : an **edge** that can be travelled in the Northwest or Southeast directions
        - `'X'` : a pair of overlapping diagonal directions. These represent the two separate edges connecting two separate pairs of edges.
        - `' '` : the **space** character is there only to show that no edge exists there

Output

You will submit a file (your program could output it directly, or you could paste your program's output into a file) with exactly **six** lines, the first representing a solution for the first course in the input file through the last line, which will represent a solution for the last course in the input file.

Each line should only contain a string of directional codes ('0' through '7'), with no leading or trailing whitespace… except for the newline at the end of the line :-)

As soon as the judging program encounters any other character or an attempt to make an invalid move, it will stop scoring that solution and move on to the next course and solution.

Evaluation

To find out the score your program's solution will earn, run the Python judging program with the command listed below (this command is assuming your file submission is called "solution.txt").

```
python3 turn_judge.py courses.txt solution.txt
```

GOOD LUCK!