# 11160   Going Together

You are playing a computer game in which three robots (Aneed, Ben and Cindy) are trapped in a labyrinth. Initially all three are situated in three different locations in the maze. There are three outlets through which the robots have to exit. As expected, there are several obstacles in the maze and the robots can't go through them.

The maze can be modeled as a square grid with $N \times N$ cells. The robots are placed on three different cells into the maze. You can command them to move. A single command will be activated for the three robots simultaneously. A robot will move to a new position if it is an empty cell within the maze or it is one of the target cells, otherwise the command will be ignored for that robot. Your task is to command them such a way that all of them are on three exit cells (in any order).
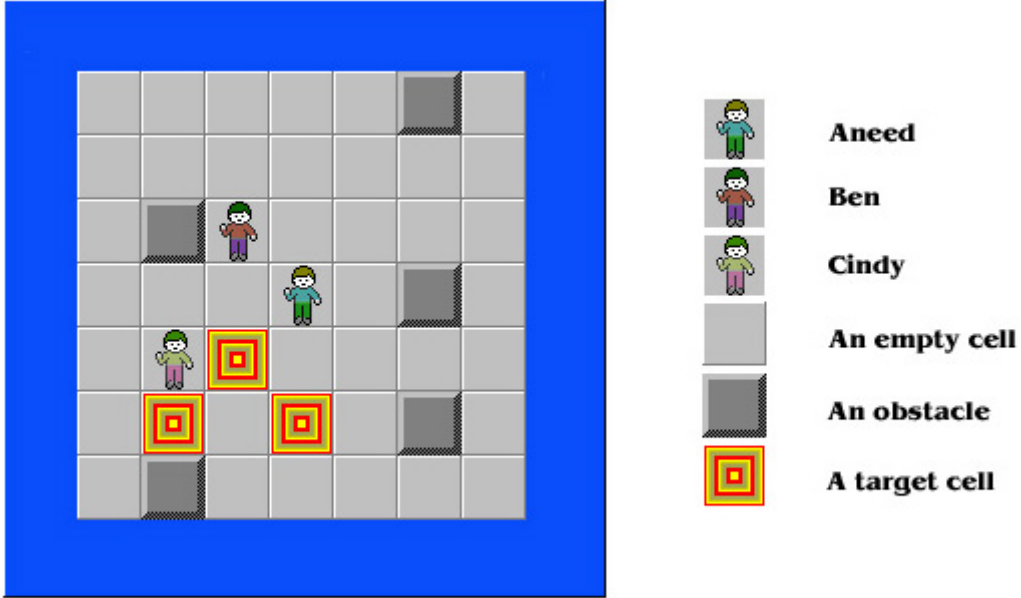
A move consists of one of the following (Each move takes 1 unit of time):

Move North   The robots move one cell north.
Move East   The robots move one cell east.
Move South   The robots move one cell south.
Move West   The robots move one cell west.

Each cell consists of one of the following characters:

  `A`   - Initial position of Aneed
  `B`   - Initial position of Ben
  `C`   - Initial position of Cindy
  `.`   - An empty cell
  `#`   - An obstacle
  `X`   - A target cell

You can assume that for every maze each of the letters (`A` `B` `C`) will appear exactly once and the letter 'X' will appear exactly 3 times.

Input

The first line of input is an integer, T ($T < 50$), that indicates the number of test cases. Each case starts with an integer N ($2 < N < 10$). Each of the next N lines contain N characters each that fills up the maze.

Output

For each case, output the case number followed by the minimum time required. If it is impossible to move them as described, print 'trapped' instead of the time.

Note: The first case corresponds to the above picture.

Sample Input

```
3
7
.....#.
.......
.#B....
...A.#.
.CX....
.X.X.#.
.#.....
3
ABC
...
XXX
3
ABC
###
XXX
```

Sample Output

```
Case 1: 2
Case 2: 2
Case 3: trapped
```