Tetris

We consider the following a slightly modified version of the game of Tetris. Initial state consists of:

- Tetris board, given as a binary occupancy matrix with *H* rows and *W* columns.
- One Tetris piece (shape), given as a binary occupancy matrix with *h* rows and *w* columns.
- One of 4 possible rotations *R* (0, 90, 180, or 270 degrees), by which the piece is rotated counterclockwise around its top-left corner before being placed on the board.
- Column *c* in the topmost row of the board, where top-left corner of the *rotated* piece is initially placed.

We say that the current position of a piece is *valid* if upon overlaying the piece and the board:

- The piece is fully contained within the board.
- No cell is occupied by both the piece and the board.

The piece moves downward by one row in every odd second (1, 3, 5, ...). If the new position is invalid, the game ends, and the player's score is the number of rows that are fully occupied *when the piece is at its final valid state*.

Task 1

Write a program that selects a piece from a given set of available pieces, along with *c* and *R*, so as to maximize the player's score.

Task 2

To increase their score, in every even second (2, 4, 6, ...), the player is allowed to translate the piece horizontally by some number of cells. The new position of the piece, as well as all intermediate translations between the initial and the final position, must be valid.

Write a program that selects a piece from a given set of available pieces, along with *c* and *R*, and a sequence of moves performed as the piece moves downward, so as to maximize the player's score.

Constraints on input

- *H* is 20, *W* is 10.
- The maximum number of available pieces is 5.
- *h* and *w* may be different for different pieces, but they are always integers between 1 and 4, inclusive.
- Each row and column of a piece contains at least one occupied cell.
- Each piece is a single connected component without "holes". Formally, if cells of a piece matrix are viewed as a 4-way connected graph
  - The subgraph induced by occupied cells is connected.
  - The subgraph induced by unoccupied cells has a "boundary cell" in each

connected component. A "boundary cell" is a cell in first or last row or column.

- On the initial board, the top 4 rows are unoccupied, guaranteeing at least one valid initial position for each piece.

Input format

- Input text file contains a number of text blocks.
- Each text block consisting of a number of consecutive nonempty lines of text.
- Every two consecutive blocks are separated by a single blank line.
- The first block encodes the board. It consists of *H* lines, each with *W* characters.
- All other blocks encode available pieces. Each such block consists of *h* lines, each with *w* characters, where *h* and *w* are the height and the width of current piece.
- Each character is either space, denoting an unoccupied cell, or a hash (#), denoting an occupied cell.
- Each line of text ends with a new line character, including the last one.

Output format

Your program should produce two lines of text, the first one containing solution to Task 1, and the second one containing solution to Task 2. Each solution is represented by the following sequence of integers separated by spaces:

- Index of the selected piece. Indexing is 0-based, and given by the order in the input file.
- Value of *R* in degrees (0, 90, 180, or 270).
- 0-based value of *c*.
- If solving Task 2, a sequence of translations for each step. Each translation is the difference between old and new positions (for example, -2 denotes translation by 2 cells to the left). **Note that one translation value is expected for each even second starting from second 2, and ending with the second that precedes the odd second in which the game ends (including those two endpoints). If any of the translation values are equal to zero, they should also be output. Solutions with incorrect number of translation values are considered invalid.**

Samples

The public dataset can be found on here. Several sample inputs are in public\set. Their expected outputs are in public\outputs.

Invocation

Your program will be invoked from command line as follows

python tetris.py

Path to input text file is given through standard input. Your program is expected to print its solution to standard output.