Bricks

Josefine is playing a tetris like game called bricks. The game takes place in a rectangular grid with 6 columns $\times$ 8 rows. A *brick* takes up a 1×1 slot in the grid. Initially the grid is empty. A *brick formation* is a rectangle where some parts are filled with bricks and the rest is air. The following is an example of a 4×3 brick formation where # represents bricks and _ represents air:

```
#_##
##__
#__#
```

The game takes place in N rounds. In each round, the player is shown a brick formation that she must decide where (horisontally) to *drop* from the top of the grid. When dropping a brick formation, each brick will indepedently fall down in a vertical line, and land either on the bottom of the grid or directly on top of another brick (from the same formation or from earlier rounds). Since the bricks fall indepedently, there will be no air holes between bricks in a column afterwards (this is unlike tetris). Before dropping the brick formation, the player may rotate it 0, 90, 180, or 270 degrees. The brick formation must be dropped such that all bricks land within the grid.

In the end of each round, all columns in the grid with at least 3 bricks will collapse and the bricks are thereby removed from the grid. A round i has an associated *round score* s_i. Let b_i be the number of collapsed bricks in a round i, the player then gets $b_i \cdot s_i$ points in that round.

The goal of the game is to maximize the score over all rounds (ie. maximize $\sum_{i=1}^{N} b_i s_i$). Help Josefine by writting a program that given the N brick formations and round scores computes the maximum possible score one can get.

Input

- Line 1: The number of rounds N ($1 \leq N \leq 300$).
- For each round:
  - Next line: The integers w_i, h_i, s_i where $w_i \times h_i$ is the dimensions of the ith brick formation, and s_i is the round score ($1 \leq w_h, h_i \leq 6$ and $0 \leq s_i \leq 10000$).
  - Next h_i lines: The brick formation represented as a $w_i \times h_i$ rectangle of #'s and _'s where # represent bricks and _ represent air. (the rectangle will always be the smallest possible rectangle that covers all bricks in the formation)

Output

- Line 1: The maximum possible score.

Example

Consider the following input with 3 rounds:

```
3
2 2 10
#_
##
3 2 4
#_#
_#_
3 3 2
#_#
###
#__
```

If we simply drop the first brick formation as long to the left as possible without rotating it we get:

```
_______
_______
_______
_______
_______
_______
#______
##_____
```

If we then rotate the second brick formation 90 degrees counter clockwise. And drop it as long to the left as possible we get: (Xs marking collapsed bricks - they will be gone when the next round starts).

```
_______
_______
_______
_______
X______
X______
X#_____
X#_____
```

Since the round score in round 2 is 4, we obtain $4 \cdot 4 = 16$ points from this. Finally, we rotate the last brick formation 180 degrees, and drop it second most to the left, we get:

```
_______
_______
_______
_______
_X_____
_X_X___
_X_X___
_X#X___
```

The last round score is 2 and thus we obtain $2 \cdot 7 = 14$ points in this round. In total we got $0 + 16 + 14 = 30$ points. This is optimal.

Scoring

Your solution will be tested on a set of test case groups. To get the points for a group, you need to pass all the test cases in the group.

#	Points	Constraints
1	30	$n \leq 5$
2	70	*No further constraints.*

Limits

Time limit: 1s for Java/C/C++ / 5s for Python
Stack limit: 100MB
Heap limit: 1000MB