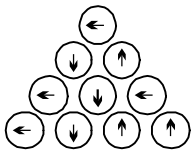
Problem	BEER	FRIENDS	GAME
Executable file	BEER.EXE	FRIENDS.EXE	GAME.EXE
Source file	BEER.BAS BEER.PAS BEER.C BEER.CPP	FRIENDS.BAS FRIENDS.PAS FRIENDS.C FRIENDS.CPP	GAME.BAS GAME.PAS GAME.C GAME.CPP
Input file	BEER.IN	FRIENDS.IN	GAME.IN
Output file	BEER.OUT	FRIENDS.OUT	GAME.OUT
Time limit per test	10 seconds	10 seconds	10 seconds
Number of tests	10	10	10
Points per test	5	7	8
Total points	**50**	**70**	**80**
	200		

BEER

Dave is preparing his birthday party. Among other things, he brought non-alcoholic beer in bottles and put them in his refrigerator. He arranged them the following way: He put as much bottles he could in one row at bottom of the refrigerator. Then he put one bottle less on the bottles from the first row, such that each new bottle lies firmly on two bottles from the first row (thus making the second row). He continued to put bottles that way until he put only one bottle to the last row (see the picture below). He arranged the bottles so that all stopples faced him.



There is an arrow drawn on each stopple. An arrow on a stopple can point to one of the following four directions: up, right, down and left.

Dave tried to turn the bottles so that all arrows on the stopples point upwards. He tried to turn bottles and realized that turning one bottle will also turn some of the neighbouring bottles.

Write a program that will for given arrangement of arrows on stopples generate a sequence of simple turns that will help Dave to make all arrows point upwards. One simple turn consist of turning a bottle 90 degrees **clockwise**. If there is a bottle up left or up right of turned bottle, it will be turned 90 degrees **counterclockwise** at the same time. Therefore, one simple turn will change directions of three bottles at most.

Your program should for given directions of arrows generate **any** sequence of simple turns which will make all arrows point upwards. A pair of numbers determines a position of a bottle: the first number denotes a row containing the bottle, and the second number denotes the position of the bottle in the row. Rows are numbered from the bottom to the top (the lowest row is numbered by 1), and bottles in a row are numbered from left to right (the leftmost bottle is numbered by 1).

Each simple turn is determined by a position of a bottle that Dave needs to turn. Total number of simple turns should not exceed 1000.

Input data

The first line of input file contains an integer N, $1 \leq N \leq 10$, the number of bottles in the lowest row.

Each of the following N lines contains a sequence of letters denoting directions of arrows on stopples in the corresponding row. Letter **U** stands for up, **L** for left, **D** for down and **R** for right.

Output data

The output file should contain a sequence of simple moves that solve the problem, one per line.

Two numbers, representing one simple move (a position of a bottle Dave needs to turn), should be written separated by a space character. The first number denotes a row containing the bottle and the second number stands for a position of that bottle.

Note: a solution needs not to be unique.

Examples

BEER.IN

```
2
U
DR
```

BEER.OUT

```
2 1
1 1
1 2
1 2
1 1
1 2
```

BEER.IN

```
3
D
DR
LLD
```

BEER.OUT

```
2 2
1 2
1 3
1 1
1 3
2 2
```

BEER.IN

```
4
L
DU
LDL
LDUU
```

BEER.OUT

```
2 3
3 2
1 2
3 1
1 2
3 1
1 1
```

FRIENDS

Clarke is a successful owner of a small company. Due to the dynamics of a work, each employee got a mobile phone to talk with other employees of the same company. Clark pays all the mobile phone bills of his employees and he tries to cut them down.

He found out about a service allowing lowering phone bills under certain conditions. Two persons that often call each other can use 'friends' service, offered by a phone company, which makes their mutual conversation cheaper. Of course, each person can be a member of only one 'friends' pair and the price of talking with others stay the same.

Clever Clarke obtained a list of calls (containing who talked with whom and how long) his employees made last month. He then tried to pair his employees to make total phone bills as small as possible.

Write a program that will help Clarke to lower his company's phone bills as much as possible using prices of 'regular' and 'friends' phone calls and list of calls of his employees.

Input data

The first line of input file contains two integers F and R, separated by a space character, $1 \leq F \leq R \leq 100$. Number F determines a price of one-minute conversation of 'friends' pair, while number R denotes a price of one minute of 'regular' conversation (i.e. of people who are not in the same 'friends' pair).

The second line contains even integer N, $2 \leq N \leq 16$, the number of employees.

The third line of input file contains an integer C, $1 \leq C \leq 10000$, the number of calls.

Each of the following C lines contain three integers, separated by a space character, A, B, M, $1 \leq A \leq N$, $1 \leq B \leq N$, $1 \leq M \leq 100$, meaning that employees A and B made a call with duration M minutes.

Output data

The first line of output file should contain the smallest possible total payment for phone bills for given list.

The next N/2 lines should contain pairs of 'friends' in any order, two numbers separated with a space character.

Note: a solution needs not to be unique.

Examples

FRIENDS.IN

```
1 2
4
4
2 3 18
2 4 26
2 3 2
1 4 12
```

FRIENDS.OUT

```
84
1 4
2 3
```

FRIENDS.IN

```
1 2
8
5
5 3 14
5 6 66
7 8 72
5 7 99
6 1 17
```

FRIENDS.OUT

```
398
1 2
3 4
5 6
7 8
```

FRIENDS.IN

```
3 10
6
4
1 3 50
3 5 85
4 1 87
2 3 73
```

FRIENDS.OUT

```
1746
1 4
2 6
3 5
```

GAME

Hal and Dave are playing an interesting game on a rectangular chess like board consisting of squares arranged in R rows and C columns. Here are the rules of the game:

- There is only one piece on a board alternately moved by the players.
- One move of the piece consists of moving it to an adjacent square in one of following directions: down, right or diagonally right–down.
- Some squares on a board are 'forbidden', i.e. the piece cannot enter such squares.
- A square may contain at most one of the following: a hamburger, french fries, ice cream. A player who moves the piece to a square with a hamburger receives 1 point, with french fries 3 points and with ice cream 5 points.
- The game ends when a player who should make a move cannot make it (because piece would fall off of board or enter a forbidden square in all three directions).
- If at the end of a game both players have same number of points, then the one who cannot make a legal move lost that game.
- If at the end of a game players have different numbers of points, then the player with more points won the game.
- Both players have 0 points at the beginning of a game. Hal makes first move. The initial position of the piece is a square that is not forbidden and it does not contain any food.

Since there is a finite number of sequence of moves finishing any given game, it can be proved that for any given initial position of the piece either Dave or Hal can win no matter how the other player plays, i.e. he has a winning strategy.

A board, positions of forbidden squares, positions of squares with food and some initial positions are given. Write a program that will determine for each given initial position which player has a winning strategy.

Input data

The first line of input file contains two integers R, number of rows ($2 \leq R \leq 100$) and C, number of columns ($2 \leq C \leq 100$) of a board, separated by a space character.

Each of next R lines contains a sequence of C characters representing a corresponding row of the board. Forbidden squares are represented with '#' character. Squares containing food are represented with letters H (hamburger), F (french fries) and I (ice cream). Other squares are represented with a dot character ('**.**').

The next line contains an integer N, $1 \leq N \leq 100$, the number of given initial positions for which a program should find which player has a winning strategy.

Each of the following N lines contain two integers A ($1 \leq A \leq R$) and B ($1 \leq B \leq C$), separated with a space character, which determine a row and a column of an initial position of a piece.

Rows are numbered from up to down with number from 1 to R, and columns are numbered from left to right with numbers from 1 to C.

Output data

Output file should contain N lines. The i^{th} line should contain name of a player (i.e. **DAVE** or **HAL**) having a winning strategy for i^{th} given initial position.

Examples

GAME.IN

```
3 4
.H#.
I...
##H.
3
1 1
1 4
2 3
```

GAME.OUT

```
HAL
DAVE
HAL
```

GAME.IN

```
4 5
.#...
#.#.F
.#..F
.#...
3
3 1
3 3
1 5
```

GAME.OUT

```
HAL
HAL
HAL
```

GAME.IN

```
5 6
##..#.
..#FH#
..#..#
###...
.....I
4
2 1
5 1
1 4
1 6
```

GAME.OUT

```
HAL
HAL
DAVE
DAVE
```