Problem A: Temple of Dune

The Archaeologists of the Current Millenium (ACM) now and then discover ancient artifacts located at the vertices of regular polygons. In general it is necessary to move one sand dune to uncover each artifact. After discovering three artifacts, the archaeologists wish to compute the minimum number of dunes that must be moved to uncover all of them.

The first line of input contains a positive integer *n*, the number of test cases. Each test case consists of three pairs of real numbers giving the *x* and *y* coordinates of three vertices from a regular polygon. For each line of input, output a single integer stating the fewest vertices that such a polygon might have.

You may assume that each input case gives three distinct vertices of a regular polygon with at most 200 vertices.

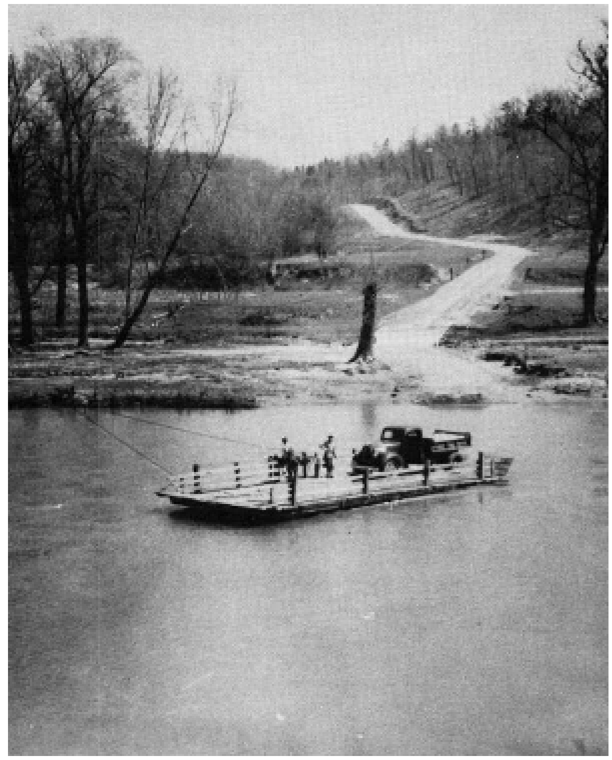
Sample input

```
4
10.00000 0.00000 0.00000 -10.00000 -10.00000 0.00000
22.23086 0.42320 -4.87328 11.92822 1.76914 27.57680
156.71567 -13.63236 139.03195 -22.04236 137.96925 -11.70517
129.400249 -44.695226 122.278798 -53.696996 44.828427 -83.507917
```

Output for the sample input

```
4
6
23
100
```

Problem B: Ferry Loading II



Before bridges were common, ferries were used to transport cars across rivers. River ferries, unlike their larger cousins, run on a guide line and are powered by the river's current. Cars drive onto the ferry from one end, the ferry crosses the river, and the cars exit from the other end of the ferry.

There is a ferry across the river that can take *n* cars across the river in *t* minutes and return in *t* minutes. *m* cars arrive at the ferry terminal by a given schedule. What is the earliest time that all the cars can be transported across the river? What is the minimum number of trips that the operator must make to deliver all cars by that time?

The first line of input contains *c*, the number of test cases. Each test case begins with *n, t, m*. *m* lines follow, each giving the arrival time for a car (in minutes since the beginning of the day). The operator can run the ferry whenever he or she wishes, but can take only the cars that have arrived up to that time. For each test case, output a single line with two integers: the time, in minutes since the beginning of the day, when the last car is delivered to the other side of the river, and the minimum number of trips made by the ferry to carry the cars within that time.

You may assume that 0 < n, t, m < 1440. The arrival times for each test case are in non-decreasing order.

Sample input

```
2
2 10 10
0
10
20
30
40
50
60
70
80
90
2 10 3
10
30
40
```

Output for sample input

```
100 5
50 2
```

Problem C: Catenyms

A catenym is a pair of words separated by a period such that the last letter of the first word is the same as the last letter of the second. For example, the following are catenyms:

```
dog.gopher
gopher.rat
rat.tiger
aloha.aloha
arachnid.dog
```

A compound catenym is a sequence of three or more words separated by periods such that each adjacent pair of words forms a catenym. For example,

```
aloha.aloha.arachnid.dog.gopher.rat.tiger
```

Given a dictionary of lower case words, you are to find a compound catenym that contains each of the words exactly once. The first line of standard input contains *t*, the number of test cases. Each test case begins with 3 <= *n* <= 1000 - the number of words in the dictionary. *n* distinct dictionary words follow; each word is a string of between 1 and 20 lowercase letters on a line by itself. For each test case, output a line giving the lexicographically least compound catenym that contains each dictionary word exactly once. Output "***" if there is no solution.

Sample Input

```
2
6
aloha
arachnid
dog
gopher
rat
tiger
3
oak
maple
elm
```

Output for Sample Input

```
aloha.arachnid.dog.gopher.rat.tiger
***
```

Problem C: Catenyms

A catenym is a pair of words separated by a period such that the last letter of the first word is the same as the last letter of the second. For example, the following are catenyms:

```
dog.gopher
gopher.rat
rat.tiger
aloha.aloha
arachnid.dog
```

A compound catenym is a sequence of three or more words separated by periods such that each adjacent pair of words forms a catenym. For example,

```
aloha.aloha.arachnid.dog.gopher.rat.tiger
```

Given a dictionary of lower case words, you are to find a compound catenym that contains each of the words exactly once. The first line of standard input contains *t*, the number of test cases. Each test case begins with 3 <= *n* <= 1000 - the number of words in the dictionary. *n* distinct dictionary words follow; each word is a string of between 1 and 20 lowercase letters on a line by itself. For each test case, output a line giving the lexicographically least compound catenym that contains each dictionary word exactly once. Output "***" if there is no solution.

Sample Input

```
2
6
aloha
arachnid
dog
gopher
rat
tiger
3
oak
maple
elm
```

Output for Sample Input

```
aloha.arachnid.dog.gopher.rat.tiger
***
```

Problem E: Rock, Scissors, Paper

Bart's sister Lisa has created a new civilization on a two-dimensional grid. At the outset each grid location may be occupied by one of three life forms: *Rocks*, *Scissors*, or *Papers*. Each day, differing life forms occupying horizontally or vertically adjacent grid locations wage war. In each war, Rocks always defeat Scissors, Scissors always defeat Papers, and Papers always defeat Rocks. At the end of the day, the victor expands its territory to include the loser's grid position. The loser vacates the position.

Your job is to determine the territory occupied by each life form after *n* days. The first line of input contains *t*, the number of test cases. Each test case begins with three integers not greater than 100: *r* and *c*, the number of rows and columns in the grid, and *n*. The grid is represented by the *r* lines that follow, each with *c* characters. Each character in the grid is R, S, or P, indicating that it is occupied by Rocks, Scissors, or Papers respectively.

For each test case, print the grid as it appears at the end of the *n*th day. Leave an empty line between the output for successive test cases.

Sample Input

```
2
3 3 1
RRR
RSR
RRR
3 4 2
RSPR
SPRS
PRSP
```

Output for Sample Input

```
RRR
RRR
RRR

RRRS
RRSP
RSPR
```