

Problem A: Complex intersecting line segments

In this problem, you are asked to determine if a set of line segments intersect.

Input

The first line of input is a number $c \leq 100$, the number of test cases.

Each test case is a single line containing four complex numbers s_1, t_1, s_2, t_2 separated by spaces. Each complex number $a + bi$ represents a point (a, b) in the xy -plane. In all cases, a and b will be integers between -100 and 100 for a complex number $a + bi$.

Note that the real part of the complex number may be omitted if it is 0 and the imaginary part may be omitted if it is 0 (and if both are 0, the number is written 0). Also, if the imaginary part is $+1$ or -1 , that part may be written as $+i$ or $-i$ (rather than $+1i$ and $-1i$). See sample input for examples.

All line segments given as input have non-zero length (i.e., the two endpoints are always different).

Output

For each test case, you should output a single line containing “intersect” if the line segment from s_1 to t_1 intersects the line segment from s_2 to t_2 (the endpoints are considered to be part of the segment) and “do not intersect” otherwise.

Sample input

```
3
2-2i -2-2i 2+2i -2+2i
1+i 2-i 2+i 1-i
0 i 1 -1+2i
```

Sample output

```
do not intersect
intersect
intersect
```

Problem B: Transformation

In a world where $1 + 1 + 1 = 2$ sits a grid of integer between 0 and 15 (inclusively). In this grid, a maximal connected component of size at least 3 containing only the number $i > 0$ transforms into a single number $i + 1 \bmod 16$. This new number is located in the bottom most row of the originating component and at the leftmost position in that row.

```
1 1 1 0 0    0 0 0 0 0    0 0 0 0 0    0 0 0 0 0
2 2 1 2 2    0 0 2 2 2    0 0 3 0 0    0 0 0 0 0
2 0 3 0 0    3 0 3 0 0    3 0 3 0 0    3 0 0 0 0
0 0 3 0 0    0 0 3 0 0    0 0 3 0 0    0 0 4 0 0
```

Numbers transform in rounds. That is, all numbers on the grid transform if they are able to. Then all numbers in the new grid transform if they are able to, and so on. This continues until no transformation is possible

The number 0 never transforms.

Input

The first line of input is a number $c \leq 20$, the number of test cases.

The first line of each test case contains two numbers $m \leq 8$ and $n \leq 8$ describing the size of the grid. The next m rows contain n numbers each. These are the initial entries of the grid.

Output

For each case, output m lines containing the state of the grid at the end of the transformation (in the same format as the input). End each case with an empty line.

Sample Input

```
2
3 3
1 1 2
2 2 3
3 3 0
4 5
1 1 1 0 0
2 2 1 2 2
2 0 3 0 0
0 0 3 0 0
```

Sample Output

```
1 1 2
2 2 3
3 3 0

0 0 0 0 0
0 0 0 0 0
3 0 0 0 0
0 0 4 0 0
```

Problem C: Chomp

We are having a chocolate eating contest. However, this is not one of those contest where the winner is the one who eats the most. No, in this contest, the winner is the contestant who is still *alive* at the end of the contest.

The bottom left corner of a $w \times h$ grid of chocolate squares has been poisoned. Contestants take turn eating a square as well all squares above and to the right of it. Of course, they must eat at least one square on their turn.

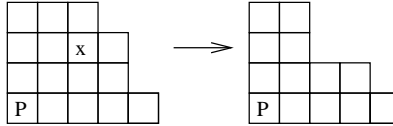


Figure 1: A move in the game. The square marked with **P** is poisonous and the square marked with **x** is the one chosen to bite out of.

Your task is to write a program to help you win the contest.

To help us, we define **winning** and **losing** positions (or configurations or shape of chocolate) recursively as follows.

- A position that contains only one square of chocolate (which must be the bottom left corner) is a losing position.
- Otherwise, a position which can reach a losing position in one move is a winning position and a position that cannot reach a losing position in one move is a losing position.

Note that position are named with respect to the player whose turn it is.

Input

The first line of input is a number $c \leq 10$, the number of test cases.

The first line of each test case is two numbers w, h with $h \leq 8, wh \leq 128$. The next line is a number $k < 100000$, the number of positions to calculate. The next k lines each consist of h numbers ℓ_1, \dots, ℓ_h where $\ell_i \leq w$ is the number of squares of chocolate in the i th row from the top. Furthermore, $\ell_i \leq \ell_{i+1}$ for all i (since these are the only positions reachable in a game).

It is now your turn.

Output

For each test case, if the position is winning, output the shape of the chocolate squares after your move (in the same format as the input). That is, output a losing position you can reach in one move. If there is more than one winning move (losing position you can reach), choose the move which lets you eat the most chocolate (of course!). If there is still a tie, choose the move which when outputted is highest in the lexicographic order.

If the input position is losing (and therefore, you are sure to lose), output the bottom left square (i.e., "0 0 0 ... 0"). Since you are going to lose, you might as well eat all the chocolate.

Sample Input

```
2
5 2
```

```
4
1 1
1 2
3 5
0 5
4 3
2
1 2 3
1 3 4
```

Sample Output

```
0 1
0 0
3 4
0 1
1 2 2
1 2 2
```

0.1 Note

40 seconds will be allowed for generating the output for this problem instead of the usual 1 second.

Problem D: Polar intersecting line segments

In this problem, you are asked to determine if a set of line segments intersect.

Input

The first line of input is a number $c \leq 20$, the number of test cases.

The first line of each test case is a number $k \leq 50000$, the number of line segments. The following k lines each consist of two numbers to eight decimal places θ_1, θ_2 with $0 < \theta_1 < 2\pi$ and $0 < \theta_2 < 2\pi$ describing the line segment from $(\cos \theta_1, \sin \theta_1)$ to $(\cos \theta_2, \sin \theta_2)$.

All line segments given as input have non-zero length (i.e., the two endpoints are always different).

Output

For each test case, you should output a single line containing “intersect” if any two of the k lines intersect and “do not intersect” otherwise. The endpoints of a line are considered to be part of the line.

Sample input

```
2
2
0 3.14159265
1.57079633 4.71238898
2
0 1.57079633
3.14159265 4.71238898
```

Sample output

```
intersect
do not intersect
```

Problem E: Letter-antiletter annihilation

Letters and antiletters can sometimes annihilate each other if they are too close. For example, the antiletter **a-** would annihilate the letter **a**, which we write as **a+** to avoid confusion. In fact, **a-** does not annihilate any other letters. Similarly, **b-** annihilates only **b+** and so on. A letter and a corresponding antiletter will annihilate each other if they are next to each other.

Given a string of letters and antiletters, we would like to know what is left after all annihilations occur.

Input

Each line will consist of a string of letters and antiletters of length at most $5 * 10^5$. Each letter or antiletter is a lower case characters followed by a symbol + or -. The input terminates with an empty line. There will be at most 30 lines in total.

Output

For each line, output the string after all annihilations have occurred. The

Sample input

```
a+b+a-b-  
a+a-  
c+a+b+b-a-d-
```

Sample output

```
a+b+c+a-b-c-  
c+d-
```

0.2 Note

40 seconds will be allowed for generating the output for this problem instead of the usual 1 second.