Coder-Strike 2014 - Qualification Round

A. Password Check

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have probably registered on Internet sites many times. And each time you should enter your invented password. Usually the registration form automatically checks the password's crypt resistance. If the user's password isn't complex enough, a message is displayed. Today your task is to implement such an automatic check.

Web-developers of the company Q assume that a password is complex enough, if it meets all of the following conditions:

- the password length is at least 5 characters;
- the password contains at least one large English letter;
- the password contains at least one small English letter;
- the password contains at least one digit.

You are given a password. Please implement the automatic check of its complexity for company Q.

Input

The first line contains a non-empty sequence of characters (at most 100 characters). Each character is either a large English letter, or a small English letter, or a digit, or one of characters: "`!`", "`?`", "`.`", "`,`", "`_`".

Output

If the password is complex enough, print message "`Correct`" (without the quotes), otherwise print message "`Too weak`" (without the quotes).

Sample test(s)

input
abacaba
output
Too weak

input
X12345
output
Too weak

input
CONTEST_is_STARTED!!11
output
Correct

B. Multi-core Processor

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The research center Q has developed a new multi-core processor. The processor consists of n cores and has k cells of cache memory. Consider the work of this processor.

At each cycle each core of the processor gets one instruction: either do nothing, or the number of the memory cell (the core will write an information to the cell). After receiving the command, the core executes it immediately. Sometimes it happens that at one cycle, multiple cores try to write the information into a single cell. Unfortunately, the developers did not foresee the possibility of resolving conflicts between cores, so in this case there is a *deadlock*: all these cores and the corresponding memory cell are locked forever. Each of the locked cores ignores all further commands, and no core in the future will be able to record an information into the locked cell. If any of the cores tries to write an information into some locked cell, it is immediately locked.

The development team wants to explore the deadlock situation. Therefore, they need a program that will simulate the processor for a given set of instructions for each core within m cycles . You're lucky, this interesting work is entrusted to you. According to the instructions, during the m cycles define for each core the number of the cycle, during which it will become locked. It is believed that initially all cores and all memory cells are not locked.

Input

The first line contains three integers n, m, k $(1 \le n, m, k \le 100)$. Then follow n lines describing instructions. The i-th line contains m integers: $x_{i1}, x_{i2}, ..., x_{im}$ $(0 \le x_{ij} \le k)$, where x_{ij} is the instruction that must be executed by the i-th core at the j-th cycle. If x_{ij} equals 0, then the corresponding instruction is «do nothing». But if x_{ij} is a number from 1 to k, then the corresponding instruction is «write information to the memory cell number x_{ij}».

We assume that the cores are numbered from 1 to n, the work cycles are numbered from 1 to m and the memory cells are numbered from 1 to k.

Output

Print n lines. In the i-th line print integer t_i. This number should be equal to 0 if the i-th core won't be locked, or it should be equal to the number of the cycle when this core will be locked.

Sample test(s)

input
4 3 5 1 0 0 1 0 2 2 3 1 3 2 0
output
1 1 3 0

input
3 2 2 1 2 1 2 2 2
output
1 1 0

input
1 1 1 0
output
0

C. Kicker

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Kicker (table football) is a board game based on football, in which players control the footballers' figures mounted on rods by using bars to get the ball into the opponent's goal. When playing two on two, one player of each team controls the goalkeeper and the full-backs (plays defence), the other player controls the half-backs and forwards (plays attack).

Two teams of company Q decided to battle each other. Let's enumerate players from both teams by integers from 1 to 4. The first and second player play in the first team, the third and the fourth one play in the second team. For each of the four players we know their game skills in defence and attack. The defence skill of the i-th player is a_i, the attack skill is b_i.

Before the game, the teams determine how they will play. First the players of the first team decide who will play in the attack, and who will play in the defence. Then the second team players do the same, based on the choice of their opponents.

We will define a team's defence as the defence skill of player of the team who plays defence. Similarly, a team's attack is the attack skill of the player of the team who plays attack. We assume that one team is guaranteed to beat the other one, if its defence is strictly greater than the opponent's attack and its attack is strictly greater than the opponent's defence.

The teams of company Q know each other's strengths and therefore arrange their teams optimally. Identify the team that is guaranteed to win (if both teams act optimally) or tell that there is no such team.

Input

The input contain the players' description in four lines. The i-th line contains two space-separated integers a_i and b_i $(1 \le a_i, b_i \le 100)$ — the defence and the attack skill of the i-th player, correspondingly.

Output

If the first team can win, print phrase "`Team 1`" (without the quotes), if the second team can win, print phrase "`Team 2`" (without the quotes). If no of the teams can definitely win, print "`Draw`" (without the quotes).

Sample test(s)

input
1 100 100 1 99 99 99 99
output
Team 1

input
1 1 2 2 3 3 2 2
output
Team 2

input
3 3 2 2 1 1 2 2
output
Draw

Note

Let consider the first test sample. The first team can definitely win if it will choose the following arrangement: the first player plays attack, the second player plays defence.

Consider the second sample. The order of the choosing roles for players makes sense in this sample. As the members of the first team choose first, the members of the second team can beat them (because they know the exact defence value and attack value of the first team).