Problem A. Artificial Intelligence?

Input file:	`ai.in`
Output file:	`ai.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Physics teachers in high school often think that problems given as text are more demanding than pure computations. After all, the pupils have to read and understand the problem first!

So they don't state a problem like "$U = 10V$, $I = 5A$, $P =$?" but rather like "You have an electrical circuit that contains a battery with a voltage of $U = 10V$ and a light-bulb. There's an electrical current of $I = 5A$ through the bulb. Which power is generated in the bulb?".

However, half of the pupils just don't pay attention to the text anyway. They just extract from the text what is given: $U = 10V$, $I = 5A$. Then they think: "Which formulae do I know? Ah yes, $P = U * I$. Therefore $P = 10V * 5A = 500W$. Finished."

OK, this doesn't always work, so these pupils are usually not the top scorers in physics tests. But at least this simple algorithm is usually good enough to pass the class. (Sad but true.)

Today we will check if a computer can pass a high school physics test. We will concentrate on the *P-U-I* type problems first. That means, problems in which two of power, voltage and current are given and the third is wanted.

Your job is to write a program that reads such a text problem and solves it according to the simple algorithm given above.

Input

The first line of the input file will contain the number of test cases.

Each test case will consist of one line containing exactly two data fields and some additional arbitrary words. A data field will be of the form $I = xA$, $U = xV$ or $P = xW$, where x is a real number. Directly before the unit (A, V or W) one of the prefixes m (milli), k (kilo) and M (Mega) may also occur.

To summarize it — data fields adhere to the following grammar:

```
DataField ::= Concept '=' RealNumber [Prefix] Unit
Concept ::= 'P' | 'U' | 'I'
Prefix ::= 'm' | 'k' | 'M'
Unit ::= 'W' | 'V' | 'A'
```

Additional assertions:

- The equal sign ('=') will never occur in an other context than within a data field.

- There is no whitespace (tabs, blanks) inside a data field.

- Either P and U, P and I, or U and I will be given.

Output

For each test case, print three lines:

- a line saying "Problem #k" where k is the number of the test case

- a line giving the solution (voltage, power or current, dependent on what was given), written without a prefix and with two decimal places as shown in the sample output

- a blank line

Sample input and output

ai.in
3
If the voltage is U=200V and the current is I=4.5A, which power is generated?
A light-bulb yields P=100W and the voltage is U=220V. Current, please.
bla bla bla lightning strike I=2A bla bla bla P=2.5MW bla bla voltage?

ai.out
Problem #1
P=900.00W
Problem #2
I=0.45A
Problem #3
U=1250000.00V

Problem B. AntiQuickSort

Input file:	`antiqs.in`
Output file:	`antiqs.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

QuickSort is routinely used to sort large arrays of numbers. Here's one implementation of QuickSort:

```
void QSort(int* a, int left, int right) {
  int key = a[(left + right) / 2];
  int i = left;
  int j = right;
  do {
    while (a[i] < key) ++i;
    while (key < a[j]) --j;
    if (i <= j) {
      int buf = a[i];
      a[i] = a[j];
      a[j] = buf;
      ++i;
      --j;
    }
  } while (i <= j);
  if (left < j) QSort(a, left, j);
  if (i < right) QSort(a, i, right);
}
...
QSort(a, 0, n - 1);
```

While QuickSort is the fastest sorting algorithm on average, there exist arrays that make it work for very long time. We will define the *execution time* of this algorithm to be the total number of comparisons made between the elements of the array (in the two inner "while" loops).

You need to create a permutation of numbers between 1 and n that will maximize the execution time.

Input

The only line of the input file will contain one integer n, $1 \leq n \leq 70000$.

Output

Output a space-separated permutation of numbers between 1 and n that maximizes the execution time of the above QuickSort algorithm. If there are several such permutations, output any.

Sample input and output

antiqs.in	antiqs.out
3	1 3 2

Problem C. The Settlers of Catan

Input file:	`catan.in`
Output file:	`catan.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Within Settlers of Catan, the 1995 German game of the year, players attempt to dominate an island by building roads, settlements and cities across its uncharted wilderness. You are employed by a software company that just has decided to develop a computer version of this game, and you are chosen to implement one of the game's special rules:

When the game ends, the player who built the longest road gains two extra victory points.

The problem here is that the players usually build complex road networks and not just one linear path. Therefore, determining the longest road is not trivial (although human players usually see it immediately).

Compared to the original game, we will solve a simplified problem here: You are given a set of nodes (cities) and a set of edges (road segments) of length 1 connecting the nodes. The longest road is defined as the longest path within the network that doesn't use an edge twice. Nodes may be visited more than once, though.

Input

The input file will contain one or more test cases. The first line of each test case contains two integers: the number of nodes n ($2 \le n \le 25$) and the number of edges m ($1 \le m \le 25$). The next m lines describe the m edges. Each edge is given by the numbers of the two nodes connected by it. Nodes are numbered from 0 to $n-1$. Edges are undirected. Nodes have degrees of three or less. The network is not neccessarily connected. Input will be terminated by two values of 0 for n and m.

Output

For each test case, print the length of the longest road on a single line.

Sample input and output

catan.in	catan.out
3 2	2
0 1	12
1 2	
15 16	
0 2	
1 2	
2 3	
3 4	
3 5	
4 6	
5 7	
6 8	
7 8	
7 9	
8 10	
9 11	
10 12	
11 12	
10 13	
12 14	
0 0	

Problem D. Error Correction

Input file:	`error.in`
Output file:	`error.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

A boolean matrix has the *parity property* when each row and each column has an even sum, i.e. contains an even number of bits which are set. Here's a 4×4 matrix which has the parity property:

```
1 0 1 0
0 0 0 0
1 1 1 1
0 1 0 1
```

The sums of the rows are 2, 0, 4 and 2. The sums of the columns are 2, 2, 2 and 2.

Your job is to write a program that reads in a matrix and checks if it has the parity property. If not, your program should check if the parity property can be established by changing only one bit. If this is not possible either, the matrix should be classified as *corrupt*.

Input

The input file will contain one or more test cases. The first line of each test case contains one integer n ($n < 100$), representing the size of the matrix. On the next n lines, there will be n integers per line. No other integers than 0 and 1 will occur in the matrix. Input will be terminated by a value of 0 for n.
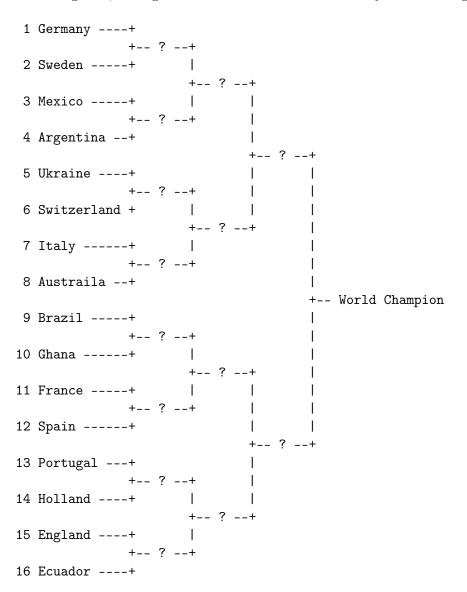
Output

For each matrix in the input file, print one line. If the matrix already has the parity property, print "OK". If the parity property can be established by changing one bit, print "Change bit (i,j)" where i is the row and j the column of the bit to be changed. Otherwise, print "Corrupt".

Sample input and output

error.in	error.out
4	OK
1 0 1 0	Change bit (2,3)
0 0 0 0	Corrupt
1 1 1 1	
0 1 0 1	
4	
1 0 1 0	
0 0 1 0	
1 1 1 1	
0 1 0 1	
4	
1 0 1 0	
0 1 1 0	
1 1 1 1	
0 1 0 1	
0	

Problem E. Germany' 06

Input file:	germany06.in
Output file:	germany06.out
Time limit:	2 seconds
Memory limit:	64 megabytes

The first round of the Soccer World Championship in Germany is coming to an end. 16 countries are remaining now, among which the winner is determined by the following tournament:

```
 1 Germany ----+
               +-- ? --+
 2 Sweden -----+       |
                       +-- ? --+
 3 Mexico -----+       |       |
               +-- ? --+       |
 4 Argentina --+               |
                               +-- ? --+
 5 Ukraine ----+               |       |
               +-- ? --+       |       |
 6 Switzerland +       |       |       |
                       +-- ? --+       |
 7 Italy ------+       |               |
               +-- ? --+               |
 8 Austraila --+                       |
                                       +-- World Champion
 9 Brazil -----+                       |
               +-- ? --+               |
10 Ghana ------+       |               |
                       +-- ? --+       |
11 France -----+       |       |       |
               +-- ? --+       |       |
12 Spain ------+               |       |
                               +-- ? --+
13 Portugal ---+               |
               +-- ? --+       |
14 Holland ----+       |       |
                       +-- ? --+
15 England ----+       |
               +-- ? --+
16 Ecuador ----+
```

For each possible match A vs. B between these 16 nations, you are given the probability that team A wins against B. This (together with the tournament mode displayed above) is sufficient to compute the probability that a given nation wins the World Cup. For example, if Italy wins against Australia with 80%, Ukraine against Switzerland with 60%, Italy against Ukraine with 70% and Italy against Switzerland with 90%, then the probability that Italy reaches the semi-finals is 80% * (70% * 60% + 90% * 40%) = 62.4%.

Your task is to write a program that computes the chances of the 16 nations to become the World Champion '06.

Input
The input file will contain just one test case.

The first 16 lines of the input file give the names of the 16 countries, from top to bottom according to the picture given above.

Next, there will follow a 16×16 integer matrix P where element p_{ij} gives the probability in percent that country i defeats country j in a direct match. Country i means the i-th country from top to bottom given in the list of countries. In the picture above Germany is number 1 and Portugal is number 13, so $p_{1,13} = 55$ would mean that in a match between Germany and Portugal, Germany wins with a probability of 55%. Note that matches may not end with a draw, i.e. $p_{ij} + p_{ji} = 100$ for all i,j.

Output

Output 16 lines of the form "XXXXXXXXXX p=Y.YY%", where XXXXXXXXXX is the country's name, left-justified in a field of 10 characters, and Y.YY is their chance in percent to win the cup, written to two decimal places. Use the same order of countries like in the input file.

Sample input and output

germany06.in
Brazil
Chile
Nigeria
Denmark
Holland
Yugoslavia
Argentina
England
Italy
Norway
France
Paraguay
Germany
Mexico
Romania
Croatia
50 65 50 60 55 50 50 65 45 55 40 55 40 55 50 50
35 50 35 45 40 35 35 50 30 40 25 40 25 40 35 35
50 65 50 60 55 50 50 65 45 55 40 55 40 55 50 50
40 55 40 50 45 40 40 55 35 45 30 45 30 45 40 40
45 60 45 55 50 45 45 60 40 50 35 50 35 50 45 45
50 65 50 60 55 50 50 65 45 55 40 55 40 55 50 50
50 65 50 60 55 50 50 65 45 55 40 55 40 55 50 50
35 50 35 45 40 35 35 50 30 40 25 40 25 40 35 35
55 70 55 65 60 55 55 70 50 60 45 60 45 60 55 55
45 60 45 55 50 45 45 60 40 50 35 50 35 50 45 45
60 75 60 70 65 60 60 75 55 65 50 65 50 65 60 60
45 60 45 55 50 45 45 60 40 50 35 50 35 50 45 45
60 75 60 70 65 60 60 75 55 65 50 65 50 65 60 60
45 60 45 55 50 45 45 60 40 50 35 50 35 50 45 45
50 65 50 60 55 50 50 65 45 55 40 55 40 55 50 50
50 65 50 60 55 50 50 65 45 55 40 55 40 55 50 50
germany06.out
Brazil   p=8.54%
Chile   p=1.60%
Nigeria   p=8.06%
Denmark   p=2.79%
Holland   p=4.51%
Yugoslavia p=7.50%
Argentina p=8.38%
England   p=1.56%
Italy   p=9.05%
Norway   p=3.23%
France   p=13.72%
Paraguay   p=3.09%
Germany   p=13.79%
Mexico   p=3.11%
Romania   p=5.53%
Croatia   p=5.53%

Problem F. Goldbach's Conjecture

Input file:	goldbach.in
Output file:	goldbach.out
Time limit:	3 seconds
Memory limit:	64 megabytes

In 1742, Christian Goldbach, a German amateur mathematician, sent a letter to Leonhard Euler in which he made the following conjecture:

Every even number greater than 4 can be written as the sum of two odd prime numbers.

For example:

- $8 = 3 + 5$. Both 3 and 5 are odd prime numbers.

- $20 = 3 + 17 = 7 + 13$.

- $42 = 5 + 37 = 11 + 31 = 13 + 29 = 19 + 23$.

Today it is still unproven whether the conjecture is right. (Oh wait, I have the proof of course, but it is too long to write it on the margin of this page.)

Anyway, your task is now to verify Goldbach's conjecture for all even numbers less than a million.

Input

Thenput file will contain one or more test cases. Each test case consists of one even integer n with $6 \leq n < 1000000$. Input will be terminated by a value of 0 for n.

Output

For each test case, print one line of the form $n = a + b$, where a and b are odd primes. Numbers and operators should be separated by exactly one blank like in the sample output below. If there is more than one pair of odd primes adding up to n, choose the pair where the difference $b - a$ is maximized. If there is no such pair, print a line saying "Goldbach's conjecture is wrong."

Sample input and output

goldbach.in	goldbach.out
8	8 = 3 + 5
20	20 = 3 + 17
42	42 = 5 + 37
0	

Problem G. Nearest number

Input file:	`nearnum.in`
Output file:	`nearnum.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Consider an $n \times n$ matrix A of non-negative integers. We define the *distance* between two its elements A_{ij} and A_{pq} to be $|i - p| + |j - q|$.

For each cell that has a zero in it, you should replace that zero with the value from the closest non-zero cell. However, if there are two or more closest non-zero cells, you should keep the zero.

Input

The first line of the input file will contain one integer n ($1 \le n \le 200$).

The next n lines will contain n space-separated integers each, describing the matrix A, $0 \le A_{ij} \le 1000000$.

Output

Output n lines with n space-separated integers each, describing the updated matrix.

Sample input and output

nearnum.in	nearnum.out
3	1 0 2
0 0 0	1 0 2
1 0 2	0 3 0
0 3 0	

Problem H. Team Queue

Input file:	`team.in`
Output file:	`team.out`
Time limit:	3 seconds
Memory limit:	64 megabytes

Queues and *Priority Queues* are data structures which are known to most computer scientists. The *Team Queue*, however, is not so well known, though it occurs often in everyday life. At lunch time the queue in front of the Mensa is a team queue, for example.

In a team queue each element belongs to a team. If an element enters the queue, it first searches the queue from head to tail to check if some of its *teammates* (elements of the same team) are already in the queue. If yes, it enters the queue right behind them. If not, it enters the queue at the tail and becomes the new last element (bad luck). Dequeuing is done like in normal queues: elements are processed from head to tail in the order they appear in the team queue.

Your task is to write a program that simulates such a team queue.

Input

The input file will contain one or more test cases. Each test case begins with the number of teams t ($1 \leq t \leq 1000$). Then t team descriptions follow, each one consisting of the number of elements belonging to the team and the elements themselves. Elements are integers in the range between 0 and 999999. A team may consist of up to 1000 elements.

Finally, a list of commands follows. There are three different kinds of commands:

- ENQUEUE x - enter element x into the team queue

- DEQUEUE - process the first element and remove it from the queue

- STOP - end of test case

The input will be terminated by a value of 0 for t.

A test case may contain up to 200000 (two hundred thousand) commands.

Output

For each test case, first print a line saying "Scenario #k", where k is the number of the test case. Then, for each DEQUEUE command, print the element which is dequeued on a single line. Print a blank line after each test case, even after the last one.

Sample input and output

team.in	team.out
2	Scenario #1
3 101 102 103	101
3 201 202 203	102
ENQUEUE 101	103
ENQUEUE 201	201
ENQUEUE 102	202
ENQUEUE 202	203
ENQUEUE 103	
ENQUEUE 203	
DEQUEUE	Scenario #2
DEQUEUE	259001
DEQUEUE	259002
DEQUEUE	259003
DEQUEUE	259004
DEQUEUE	259005
STOP	260001
2	
5 259001 259002 259003 259004 259005	
6 260001 260002 260003 260004 260005	
260006	
ENQUEUE 259001	
ENQUEUE 260001	
ENQUEUE 259002	
ENQUEUE 259003	
ENQUEUE 259004	
ENQUEUE 259005	
DEQUEUE	
DEQUEUE	
ENQUEUE 260002	
ENQUEUE 260003	
DEQUEUE	
DEQUEUE	
DEQUEUE	
DEQUEUE	
STOP	
0	