Problem A
Family

ACM Central European Programming Contest, Warsaw 2002, Poland

We want to find out how much are the members of a family of monsters related. Each monster has the same number of genes but the genes themselves may differ from monster to monster. It would be nice to know how many genes any two given monsters have in common. However this is impossible, because the number of genes is very large. Fortunately, we do know the family tree (well, not actually a tree, but you cannot really blame them, after all they are monsters, right?) and we do know how the genes are inherited so we can estimate the number of common genes quite well.

The inheritance rule is very simple: if a monster C is a child of monsters A and B then each gene of C is identical to the corresponding gene of either A or B, each with probability 50%. Every gene of every monster is inherited independently.

Let us define the degree of relationship of monsters X and Y as the expected number of common genes. For example consider a family consisting of two completely unrelated (i.e. having no common genes) monsters A and B and their two children C and D. How much are C and D related? Well, each of C's genes comes either from A or from B, both with probability 50%. The same is true for D. Thus, the probability of a given gene of C being the same as the corresponding gene of D is 50%. Therefore the degree of relationship of C and D (the expected number of common genes) is equal to 50% of all the genes. Note that the answer would be different if A and B were related. If A and B have common genes they will be always inherited by both C and D.

Your task is to write a program that, when given a family graph and a list of pairs of monsters, computes the degree of relationship for each of these pairs.

Task

Write a program that:

- reads the description of a family and a list of pairs of its members from the standard input,
- computes the degree of relationship (in percentages) for each pair on the list,
- writes the result to the standard output.

Input

The first line of the input contains two integers n and k separated by a single space. Integer n ($2 \le n \le 300$) is the number of members of a family. Family members are indexed arbitrarily from 1 to n. Integer k ($0 \le k \le n-2$) is the number of monsters that do have parents (all the other monsters were created by Gods and are completely unrelated to each other).

Each of the next k lines contains three different integers a, b, c separated by single spaces. The triple a, b, c means that the monster a is a child of monsters b and c.

The next input line contains an integer m ($1 \le m \le n^2$) — the number of pairs of monsters on the list. Each of the next m lines contains two integers separated by a single space — these are the indices of two monsters.

You may assume that no monster is its own ancestor. You should not make any additional assumptions on the input data. In particular, you should not assume that there exists any valid sex assignment.

Output

The output consists of m lines. The i-th line corresponds to the i-th pair on the input list and should contain single number followed by the percentage sign. The number should be the exact degree of

relationship (in percentages) of the monsters in the i-th pair. Unsignificant zeroes are not allowed in the output (please note however that there must be at least one digit before the period sign so for example the leading zero in number 0.1 is significant and you cannot print it as .1). Confront the example output for the details of the output format.

Example

For the input:

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

the correct answer is:

```
0%
50%
81.25%
100%
```

Problem B
Intervals

ACM Central European Programming Contest, Warsaw 2002, Poland

You are given n closed, integer intervals $[a_i, b_i]$ and n integers $c_1, \ldots, c_n$.

Task

Write a program that:

- reads the number of intervals, their endpoints and integers $c_1, \ldots, c_n$ from the standard input,
- computes the minimal size of a set Z of integers which has at least c_i common elements with interval $[a_i, b_i]$, for each $i = 1, 2, \ldots, n$,
- writes the answer to the standard output.

Input

The first line of the input contains an integer n ($1 \leq n \leq 50\ 000$) — the number of intervals. The following n lines describe the intervals. The line $i + 1$ of the input contains three integers a_i, b_i, c_i separated by single spaces and such that $0 \leq a_i \leq b_i \leq 50\ 000$ and $1 \leq c_i \leq b_i - a_i + 1$.

Output

The output contains exactly one integer equal to the minimal size of a set Z sharing at least c_i elements with interval $[a_i, b_i]$, for each $i = 1, 2, \ldots, n$.

Example

For the input:

```
5
3 7 3
8 10 3
6 8 1
1 3 1
10 11 1
```

the correct answer is:

```
6
```

Problem C
One-way traffic

ACM Central European Programming Contest, Warsaw 2002, Poland

In a certain town there are n intersections connected by two- and one-way streets. The town is very modern so a lot of streets run through tunnels or viaducts. Of course it is possible to travel between any two intersections in both ways, i.e. it is possible to travel from an intersection a to an intersection b as well as from b to a without violating traffic rules. Because one-way streets are safer, it has been decided to create as much one-way traffic as possible. In order not to make too much confusion it has also been decided that the direction of traffic in already existing one-way streets should not be changed.

Your job is to create a new traffic system in the town. You have to determine the direction of traffic for as many two-way streets as possible and make sure that it is still possible to travel both ways between any two intersections.

Task

Write a program that:

- reads a description of the street system in the town from the standard input,
- for each two-way street determines one direction of traffic or decides that the street must remain two-way,
- writes the answer to the standard output.

Input

The first line of the input contains two integers n and m, where $2 \leq n \leq 2000$ and $n-1 \leq m \leq n(n-1)/2$. Integer n is the number of intersections in the town and integer m is the number of streets.

Each of the next m lines contains three integers a, b and c, where $1 \leq a \leq n$, $1 \leq b \leq n$, $a \neq b$ and c belongs to $\{1, 2\}$. If $c = 1$ then intersections a and b are connected by an one-way street from a to b. If $c = 2$ then intersections a and b are connected by a two-way street. There is at most one street connecting any two intersections.

Output

The output contains exactly the same number of lines as the number of two-way streets in the input. For each such street (in any order) the program should write three integers a, b and c meaning, the new direction of the street from a to b ($c = 1$) or that the street connecting a and b remains two-way ($c = 2$). If there are more than one solution with maximal number of one-way streets then your program should output any of them but just one.

Example

For the input:	the correct answer is:

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

Problem D. Balloons

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

As you may know, balloons are handed out during ACM contests to teams as they solve problems. However, this sometimes presents logistical challenges. In particular, one contest hosting site maintains two rooms, A and B, each containing a supply of balloons. There are N teams attending the contest at that site, each sitting at a different location. Some are closer to room A, others are closer to room B, and others are equally distant. Given the number of balloons needed by each team and the distance from each team to room A, and to room B, what is the minimum total possible distance that must be traveled by all balloons as they are delivered to their respective teams, assuming they are allocated in an optimal fashion from rooms A and B? For the purposes of this problem, assume that all of the balloons are identical.

Input

There will be several test cases in the input. Each test case will begin with a line with three integers N, A, B where N is the number of teams ($1 \leq N \leq 1000$), and A and B are the number of balloons in rooms A and B, respectively ($0 \leq A, B \leq 10000$). On each of the next N lines there will be three integers, representing information for each team: K, DA, DB, Where K is the total number of balloons that this team will need, DA is the distance of this team from room A, and DB is this team's distance from room B ($0 \leq DA, DB \leq 1000$). You may assume that there are enough balloons — that is, sum of values K is not greater than $A + B$. The input will end with a line with three 0s.

Output

For each test case, output a single integer, representing the minimum total distance that must be traveled to deliver all of the balloons. Count only the outbound trip, from room A or room B to the team. Don't count the distance that a runner must travel to return to room A or room B. Print each integer on its own line with no spaces. Do not print any blank lines between answers.

Example

standard input	standard output
3 15 35 10 20 10 10 10 30 10 40 10 0 0 0	300

Problem E
Servers

ACM Central European Programming Contest, Warsaw 2002, Poland

The Kingdom of Byteland decided to develop a large computer network of servers offering various services.

The network is built of n servers connected by bidirectional wires. Two servers can be directly connected by at most one wire. Each server can be directly connected to at most 10 other servers and every two servers are connected by some path in the network. Each wire has a fixed positive data transmission time measured in milliseconds. The distance (in milliseconds) $\delta(V, W)$ between two servers V and W is defined as the length of the shortest (according to transmission time) path connecting V and W in the network. For convenience we let $\delta(V, V) = 0$ for all V.

Some servers offer more services than others. Therefore each server V is marked with a natural number $r(V)$, called a rank. The bigger the rank the more powerful a server is.

At each server, data about nearby servers should be stored. However, not all servers are interesting. The data about distant servers with low ranks do not have to be stored. More specifically, a server W is interesting for a server V if for every server U such that $\delta(V, U) \leq \delta(V, W)$ we have $r(U) \leq r(W)$.

For example, all servers of the maximal rank are interesting for all servers. If a server V has the maximal rank, then exactly the servers of the maximal rank are interesting for V. Let $B(V)$ denote the set of servers interesting for a server V.

We want to compute the total amount of data about servers that need to be stored in the network being the total sum of sizes of all sets $B(V)$. The Kingdom of Byteland wanted the data to be quite small so it built the network in such a way that this sum does not exceed $30n$.

Task

Write a program that:

- reads the description of a server network from the standard input,
- computes the total amount of data about servers that need to be stored in the network,
- writes the result to the standard output.

Input

The first line contains two natural numbers n, m, where n is the number of servers in the network ($1 \leq n \leq 30\,000$) and m is the number of wires ($1 \leq m \leq 5n$). The numbers are separated by single space.

The ranks of the servers are given in the next n lines. Line i contains one integer $r(i)$ ($1 \leq r(i) \leq 10$) — the rank of the i-th server.

The wires are described in the next m lines. Each wire is described by three numbers a, b, t ($1 \leq t \leq 1000$, $1 \leq a, b \leq n$, $a \neq b$), where a and b are numbers of the servers connected by the wire and t is the transmission time of the wire in milliseconds.

Output

The output consists of a single integer equal to the total amount of data about servers that need to be stored in the network.

Example

For the input:

```
4 3
2
3
1
1
1 4 30
2 3 20
3 4 20
```

the correct answer is:

```
9
```

because $B(1) = \{1, 2\}$, $B(2) = \{2\}$, $B(3) = \{2, 3\}$, $B(4) = \{1, 2, 3, 4\}$.

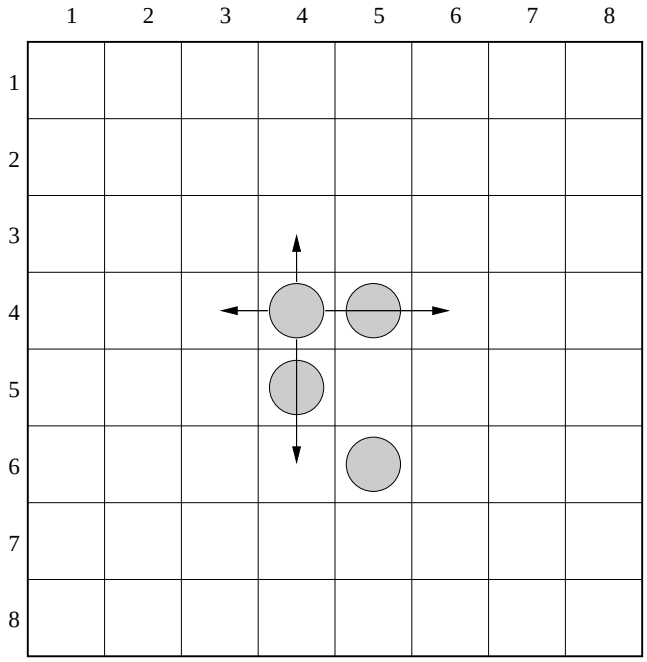
Problem F
Solitaire

ACM Central European Programming Contest, Warsaw 2002, Poland

Solitaire is a game played on a chessboard 8x8. The rows and columns of the chessboard are numbered from 1 to 8, from the top to the bottom and from left to right, respectively.

There are four identical pieces on the board. In one move it is allowed to:

- move a piece to an empty neighboring field (up, down, left or right),
- jump over one neighboring piece to an empty field (up, down, left or right).



Exactly 4 moves are allowed for each piece in the figure above. As an example let's consider a piece placed in the row 4, column 4. It can be moved one row up, two rows down, one column left or two columns right.

Task

Write a program that:

- reads two chessboard configurations from the standard input,
- verifies whether the second one is reachable from the first one in at most 8 moves,
- writes the result to the standard output.

Input

Each of two input lines contains 8 integers $a_1, a_2, \ldots, a_8$ separated by single spaces and describes one configuration of pieces on the chessboard. Integers a_{2j-1} and a_{2j} $(1 \leq j \leq 4)$ describe the position of one piece — the row number and the column number respectively.

Output

The output should contain one word YES if the configuration described in the second input line is reachable from the configuration described in the first input line in at most 8 moves, and one word NO otherwise.

Example

For the input:

```
4 4 4 5 5 4 6 5
2 4 3 3 3 6 4 6
```

the correct answer is:

```
YES
```

Problem G
Timetable

ACM Central European Programming Contest, Warsaw 2002, Poland

You are the owner of a railway system between n cities, numbered by integers from 1 to n. Each train travels from the start station to the end station according to a very specific timetable (always on time), without stopping anywhere between. A departure timetable is available on each station. Unfortunately each timetable contains only direct connections. A passenger that wants to travel from city p to city q is not limited to direct connections — he or she can change trains. Each change takes zero time, but a passenger cannot change from one train to the other if it departs before the first one arrives. People would like to have a timetable of all optimal connections. A connection departing from city p at A o'clock and arriving in city q at B o'clock is called optimal if there is no connection that begins in p not sooner than at A and ends in q not later than at B. We are only interested in connections that can be completed during the same day.

Task

Write a program that:

- reads the number of cities n and timetables from the standard input,
- creates a timetable of optimal connections from city 1 to city n,
- writes the answer to the standard output.

Input

The first line of the input contains an integer n ($2 \leq n \leq 100\ 000$). The following lines contain n timetables for cities $1, 2, \ldots, n$ respectively.

The first line of the timetable description contains only one integer m. Each of the following m lines corresponds to one position in the timetable and contains: departure time A, arrival time B ($A < B$) and destination city number t ($1 \leq t \leq n$) separated by single spaces. Departure time A and arrival time B are written in format $hh:mm$, where hh are two digits representing full hours ($00 \leq hh \leq 23$) and mm are two digits representing minutes ($00 \leq mm \leq 59$). Positions in the timetable are given in non-decreasing order according to the departure times. The number of all positions in all timetables does not exceed 1 000 000.

Output

The first line of the output contains an integer r — the number of positions in the timetable being the solution. Each of the following r lines contains a departure time A and an arrival time B separated by single space. The time format should be like in the input and positions in the timetable should be ordered increasingly according to the departure times. If there is more then one optimal connection with the same departure and arrival time, your program should output only one of them.

Example

For the input:

```
3
3
09:00 15:00 3
10:00 12:00 2
11:00 20:00 3
2
11:30 13:00 3
12:30 14:00 3
0
```

the correct answer is:

```
2
10:00 14:00
11:00 20:00
```

Problem H
Voracious Steve

ACM Central European Programming Contest, Warsaw 2002, Poland

Steve and Digit bought a box containing a number of donuts. In order to divide them between themselves they play a special game that they created. The players alternately take a certain, positive number of donuts from the box, but no more than some fixed integer. Each player's donuts are gathered on the player's side. The player that empties the box eats his donuts while the other one puts his donuts back into the box and the game continues with the "looser" player starting. The game goes on until all the donuts are eaten. The goal of the game is to eat the most donuts. How many donuts can Steve, who starts the game, count on, assuming the best strategy for both players?

Task

Write a program that:

- reads the parameters of the game from the standard input,
- computes the number of donuts Steve can count on,
- writes the result to the standard output.

Input

The first and only line of the input contains exactly two integers n and m separated by a single space, $1 \leq m \leq n \leq 100$ — the parameters of the game, where n is the number of donuts in the box at the beginning of the game and m is the upper limit on the number of donuts to be taken by one player in one move.

Output

The output contains exactly one integer equal to the number of donuts Steve can count on.

Example

For the input:

```
5 2
```

the correct answer is:

```
3
```

Problem I. Palindrometer

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

While driving the other day, John looked down at his odometer, and it read 100000. John was pretty excited about that. But, just one mile further, the odometer read 100001, and John was REALLY excited! You see, John loves palindromes — things that read the same way forwards and backwards. So, given any odometer reading, what is the least number of miles John must drive before the odometer reading is a palindrome? For John, every odometer digit counts. If the odometer reading was 000121, he wouldn't consider that a palindrome.

Input

There will be several test cases in the input. Each test case will consist of an odometer reading on its own line. Each odometer reading will be from 2 to 9 digits long. The odometer in question has the number of digits given in the input — so, if the input is 00456, the odometer has 5 digits. There will be no spaces in the input, and no blank lines between input sets. The input will end with a line with a single 0.

Output

For each test case, output the minimum number of miles John must drive before the odometer reading is a palindrome. This may be 0 if the number is already a palindrome. Output each integer on its own line, with no extra spaces and no blank lines between outputs.

Example

standard input	standard output
100000 100001 000121 00456 0	1 0 979 44

Problem J. Mine Sweeper

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

The game *Minesweeper* is played on an n by n grid. In this grid are hidden m mines, each at a distinct grid location. The player repeatedly touches grid positions. If a position with a mine is touched, the mine explodes and the player loses. If a positon not containing a mine is touched, an integer between 0 and 8 appears denoting the number of adjacent or diagonally adjacent grid positions that contain a mine. A sequence of moves in a partially played game is illustrated below.

Here, n is 8, m is 10, blank squares represent the integer 0, raised squares represent unplayed positions, and the figures resembling asterisks represent mines. The leftmost image represents the partially played game. From the first image to the second, the player has played two moves, each time choosing a safe grid position. From the second image to the third, the player is not so lucky; he chooses a position with a mine and therefore loses. The player wins if he continues to make safe moves until only m unplayed positions remain; these must necessarily contain the mines.

Your job is to read the information for a partially played game and to print the corresponding board.

Input

The first line of input contains a single postitive integer n ($n \leq 10$). The next n lines represent the positions of the mines. Each line represents the contents of a row using n characters: a period indicates an unmined positon while an asterisk indicates a mined position. The next n lines are each n characters long: touched positions are denoted by an x, and untouched positions by a period. The sample input corresponds to the middle figure above.

Output

Your output should represent the board, with each position filled in appropriately. Positions that have been touched and do not contain a mine should contain an integer between 0 and 8. If a mine has been touched, all positions with a mine should contain an asterisk. All other positions should contain a period.

Example

standard input	standard output
8 ...**..* ......*. ....*... ....*.. ...**.*. .....*.. xxx..... xxxx.... xxxx.... xxxxx... xxxxx... xxxxx... xxx..... xxxxx...	001..... 0013.... 0001.... 00011... 00001... 00123... 001..... 00123...

Problem K. Profits

Input file:	standard input
Output file:	standard output
Time limit:	4 seconds
Memory limit:	256 megabytes

Your friends have just opened a new business, and you want to see how well they are doing. The business has been running for a number of days, and your friends have recorded their net profit on each day. You want to find the largest total profit that your friends have made during any consecutive time span of at least one day. For example, if your friends' profits looked like this:

- Day 1: -3
- Day 2: 4
- Day 3: 9
- Day 4: -2
- Day 5: -5
- Day 6: 8

Their maximum profit over any span would be 14, from days 2 to 6.

Input

There will be several test cases in the input. Each test case will begin with an integer N ($1 \leq N \leq 250000$) on its own line, indicating the number of days. On each of the next N lines will be a single integer P ($-100 \leq P \leq 100$), indicating the profit for that day. The days are specified in order. The input will end with a line with a single 0.

Output

For each test case, output a single integer, representing the maximum profit over any non-empty span of time. Print each integer on its own line with no spaces. Do not print any blank lines between answers.

Example

standard input	standard output
6 -3 4 9 -2 -5 8 2 -1000 -19 0	14 -19

Problem L. Bit Counting

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 megabytes

Start with an integer, N_0, which is greater than 0. Let N_1 be the number of ones in the binary representation of N_0. So, if $N_0 = 27$, $N_1 = 4$. For all $i > 0$, let N_i be the number of ones in the binary representation of N_{i-1}. This sequence will always converge to one. For any starting number, N_0, let K be the minimum value of $i \geq 0$ for which $N_i = 1$. For example, if $N_0 = 31$, then $N_1 = 5, N_2 = 2, N_3 = 1$, so $K = 3$. Given a range of consecutive numbers, and a value X, how many numbers in the range have a K value equal to X?

Input

There will be several test cases in the input. Each test case will consist of three integers on a single line: l, r, X, where l and r ($1 \leq l \leq r \leq 10^{18}$) are the lower and upper limits of a range of integers, and X ($0 \leq X \leq 10$) is the target value for K. The input will end with a line with three 0s.

Output

For each test case, output a single integer, representing the number of integers in the range from l to r (inclusive) which have a K value equal to X in the input. Print each integer on its own line with no spaces. Do not print any blank lines between answers.

Example

standard input	standard output
31 31 3	1
31 31 1	0
27 31 1	0
27 31 2	3
1023 1025 1	1
1023 1025 2	1
0 0 0	