ACM International Collegiate Programming Contest 97/98

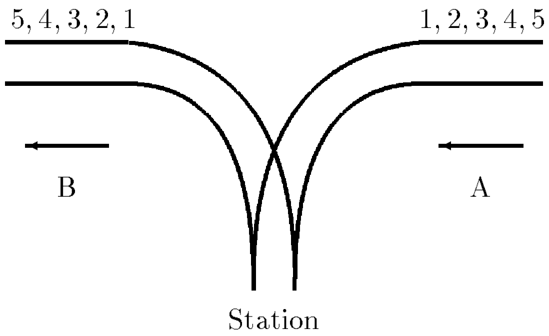
Sponsored by IBM

Central European Regional Contest

Problem A: Rails

Input file: `rails.in`
Output file: `rails.out`
Program file: `rails.pas` or `rails.cpp`

There is a famous railway station in PopPush City. Country there is incredibly hilly. The station was built in last century. Unfortunately, funds were extremely limited that time. It was possible to establish only a surface track. Moreover, it turned out that the station could be only a dead-end one (see picture) and due to lack of available space it could have only one track.



The local tradition is that every train arriving from the direction A continues in the direction B with coaches reorganized in some way. Assume that the train arriving from the direction A has $N \leq 1000$ coaches numbered in increasing order $1, 2, \ldots, N$. The chief for train reorganizations must know whether it is possible to marshal coaches continuing in the direction B so that their order will be $a_1, a_2, \ldots, a_N$. Help him and write a program that decides whether it is possible to get the required order of coaches. You can assume that single coaches can be disconnected from the train before they enter the station and that they can move themselves until they are on the track in the direction B. You can also suppose that at any time there can be located as many coaches as necessary in the station. But once a coach has entered the station it cannot return to the track in the direction A and also once it has left the station in the direction B it cannot return back to the station.

Input

The input file consists of blocks of lines. Each block except the last describes one train and possibly more requirements for its reorganization. In the first line of the block there is the integer N described above. In each of the next lines of the block there is a permutation of $1, 2, \ldots, N$. The last line of the block contains just 0.

The last block consists of just one line containing 0.

Output

The output file contains the lines corresponding to the lines with permutations in the input file. A line of the output file contains `Yes` if it is possible to marshal the coaches in the order required on the corresponding line of the input file. Otherwise it contains `No`. In addition, there is one empty line after the lines corresponding to one block of the input file. There is no line in the output file corresponding to the last "null" block of the input file.

Example

Input file:

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

```
0
6
6 5 4 3 2 1
0
0
```
Output file:
```
Yes
No

Yes
```

ACM International Collegiate Programming Contest 97/98

Sponsored by IBM

Central European Regional Contest

Problem B: King

Input file: `king.in`
Output file: `king.out`
Program file: `king.pas` or `king.cpp`

Once, in one kingdom, there was a queen and that queen was expecting a baby. The queen prayed: "If my child was a son and if only he was a sound king." After nine months her child was born, and indeed, she gave birth to a nice son.

Unfortunately, as it used to happen in royal families, the son was a little retarded. After many years of study he was able just to add integer numbers and to compare whether the result is greater or less than a given integer number. In addition, the numbers had to be written in a sequence and he was able to sum just continuous subsequences of the sequence.

The old king was very unhappy of his son. But he was ready to make everything to enable his son to govern the kingdom after his death. With regards to his son's skills he decided that every problem the king had to decide about had to be presented in a form of a finite sequence of integer numbers and the decision about it would be done by stating an integer constraint (i.e. an upper or lower limit) for the sum of that sequence. In this way there was at least some hope that his son would be able to make some decisions.

After the old king died, the young king began to reign. But very soon, a lot of people became very unsatisfied with his decisions and decided to dethrone him. They tried to do it by proving that his decisions were wrong.

Therefore some conspirators presented to the young king a set of problems that he had to decide about. The set of problems was in the form of subsequences $S_i = \{a_{s_i}, a_{s_i+1}, \ldots, a_{s_i+n_i}\}$ of a sequence $S = \{a_1, a_2, \ldots, a_n\}$. The king thought a minute and then decided, i.e. he set for the sum $a_{s_i} + a_{s_i+1} + \ldots + a_{s_i+n_i}$ of each subsequence S_i an integer constraint k_i (i.e. $a_{s_i} + a_{s_i+1} + \ldots + a_{s_i+n_i} < k_i$ or $a_{s_i} + a_{s_i+1} + \ldots + a_{s_i+n_i} > k_i$ resp.) and declared these constraints as his decisions.

After a while he realized that some of his decisions were wrong. He could not revoke the declared constraints but trying to save himself he decided to fake the sequence that he was given. He ordered to his advisors to find such a sequence S that would satisfy the constraints he set. Help the advisors of the king and write a program that decides whether such a sequence exists or not.

Input

The input file consists of blocks of lines. Each block except the last corresponds to one set of problems and king's decisions about them. In the first line of the block there are integers n, and m, where $0 < n \leq 100$ is length of the sequence S and $0 < m \leq 100$ is the number of subsequences S_i. Next m lines contain particular decisions coded in the form of quadruples s_i, n_i, o_i, k_i, where o_i represents operator $>$ (coded as `gt`) or operator $<$ (coded as `lt`) respectively. The symbols s_i, n_i, and k_i have the meaning described above. The last block consists of just one line containing `0`.

Output

The output file contains the lines corresponding to the blocks in the input file. A line contains text `successful conspiracy` when such a sequence does not exist. Otherwise it contains text `lamentable kingdom`. There is no line in the output file corresponding to the last "null" block of the input file.

Example

Input file:

```
4 2
1 2 gt 0
2 2 lt 2
```

```
1 2
1 0 gt 0
1 0 lt 0
0
```

Output file:

```
lamentable kingdom
successful conspiracy
```

ACM International Collegiate Programming Contest 97/98

Sponsored by IBM

Central European Regional Contest

Problem C: Prime Land

Input file: `primelnd.in`
Output file: `primelnd.out`
Program file: `primelnd.pas` or `primelnd.cpp`

Everybody in the Prime Land is using a prime base number system. In this system, each positive integer x is represented as follows: Let $\{p_i\}_{i=0}^{\infty}$ denote the increasing sequence of all prime numbers. We know that $x > 1$ can be represented in only one way in the form of product of powers of prime factors. This implies that there is an integer k_x and uniquely determined integers $e_{k_x}, e_{k_x-1}, \ldots, e_1, e_0$, $(e_{k_x} > 0)$, that $x = p_{k_x}^{e_{k_x}} \cdot p_{k_x-1}^{e_{k_x-1}} \cdot \ldots \cdot p_1^{e_1} \cdot p_0^{e_0}$. The sequence

$$(e_{k_x}, e_{k_x-1}, \ldots, e_1, e_0)$$

is considered to be the representation of x in prime base number system.

It is really true that all numerical calculations in prime base number system can seem to us a little bit unusual, or even hard. In fact, the children in Prime Land learn to add to subtract numbers several years. On the other hand, multiplication and division is very simple.

Recently, somebody has returned from a holiday in the Computer Land where small smart things called computers have been used. It has turned out that they could be used to make addition and subtraction in prime base number system much easier. It has been decided to make an experiment and let a computer to do the operation "minus one".

Help people in the Prime Land and write a corresponding program.

For practical reasons we will write here the prime base representation as a sequence of such p_i and e_i from the prime base representation above for which $e_i > 0$. We will keep decreasing order with regard to p_i.

Input

The input file consists of lines (at least one) each of which except the last contains prime base representation of just one positive integer greater than 2 and less or equal 32767. All numbers in the line are separated by one space. The last line contains number 0.

Output

The output file contains one line for each but the last line of the input file. If x is a positive integer contained in a line of the input file, the line in the output file will contain $x - 1$ in prime base representation. All numbers in the line are separated by one space. There is no line in the output file corresponding to the last "null" line of the input file.

Example

Input file:

```
17 1
5 1 2 1
509 1 59 1
0
```

Output file:

```
2 4
3 2
13 1 11 1 7 1 5 1 3 1 2 1
```

ACM International Collegiate Programming Contest 97/98

Sponsored by IBM

Central European Regional Contest

Problem D: Word

Input file: `word.in`
Output file: `word.out`
Program file: `word.pas` or `word.cpp`

Dr. R. E. Wright's class was studying modified L-Systems. Let us explain necessary details. As a model let us have words of length n, over a two letter alphabet $\{a, b\}$. The words are cyclic, this means we can write one word in any of n forms we receive by cyclic shift, whereby the first and the last letters in the word are considered to be neighbours.

Rewriting rules rewrite a letter at a position i, depending on letters at the positions $i-2$, i, $i+1$. We rewrite all letters of the word in one step. When we have a given starting word and a set of rewriting rules a natural question is: how does the word look after s rewriting steps?

Help Dr. R. E. Wright and write a program which solves this task.

Input

There are several blocks in the input file, each describing one system. There is an integer number n, $2 < n < 16$, the length of the input word in the first line. There is a word in the next line. The word contains only lowercase letters `a` and `b`. There are four characters $c_1c_2c_3c_4$ in the next eight lines. Each quadruple represents one rewriting rule with the following meaning: when the letter at the position $i-2$ is c_1 and the letter at the position i is c_2 and the letter at the position $i+1$ is c_3 then the letter at the position i after rewriting will be c_4. Rewriting rules are correct and complete. There is an integer number s, $0 \le s \le 2000000000$, in the last line of the block.

Output

There is one line corresponding to each block of the input file. The line contains a word which we receive after s rewriting steps from the corresponding starting word using given rewriting rules. As we mentioned above, the word can be written in any of n cyclic shifted forms. The output file contains the lexicographically smallest word, assuming that `a` < `b`.

Example

Input file:

```
5
aaaaa
aaab
aabb
abab
abbb
baab
babb
bbab
bbbb
1
```

Output file:

```
bbbbb
```

ACM International Collegiate Programming Contest 97/98

Sponsored by IBM

Central European Regional Contest

Problem E: Time

Input file: `mytime.in`
Output file: `mytime.out`
Program file: `mytime.pas` or `mytime.cpp`

Let a time point be given as 6 integers y, m, d, h, min, s, where $1970 < y < 2030$, $0 < m < 13$, $0 < d < 32$, $0 \leq h < 24$, $0 \leq min < 60$, $0 \leq s < 60$. Write a program which computes how many periods of a given length fit between two given time points. A period is given by a pair consisting of a positive integer and a word expressing a time unit, i.e. `year` or `month` or `day` or `hour` or `minute` or `second`. Every 4th year is a leap year, except every 100th which is not except every 400 year which is. A length of the year varies according to leap years. The same is true for the month February. Time units always start as usual, e.g., a year starts at 1st January, a month starts at its 1st day, a day starts at 0 hours 0 minutes 0 seconds, etc. A period ends after its last second.

Input

The input file consists of blocks of lines. Each block has three lines. The first line of a block contains a time point $D1$ and the second line a time point $D2$. $D1$ always precedes $D2$. All numbers in the lines are separated by one space. You can assume that the given descriptions of time points are correct. The third line contains a time period. There is one space between the number and the word in the period definition on this line. After each block, there is one empty line.

Output

The output file contains the lines corresponding to the blocks in the input file. A line corresponding to a block contains one integer number expressing how many specified periods are contained between the given time points.

Example

Input file:

```
1997 12 31 23 59 59
1998 1 1 0 0 0
1 second

2000 2 29 0 0 0
2000 2 29 23 59 59
1 day

2000 2 29 0 0 0
2000 3 1 0 0 0
24 hour

1996 12 31 20 30 0
1997 1 1 7 30 0
60 minute

1996 12 31 20 30 0
1997 1 1 7 30 0
1 hour
```

Output file:

```
1
0
1
11
10
```

ACM International Collegiate Programming Contest 97/98

Sponsored by IBM

Central European Regional Contest

Problem F: Puzzle

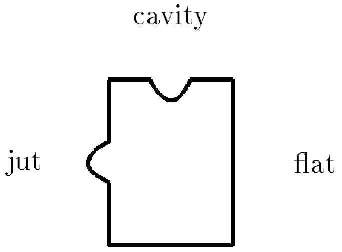
Input file: `puzzle.in`
Output file: `puzzle.out`
Program file: `puzzle.pas` or `puzzle.cpp`

Little Barborka has just started to learn how to solve a picture puzzle. She has started with a small one containing 15 pieces. Her daddy tries to solve the puzzle too. To make it a little bit harder for himself, he has turned all puzzle pieces upside down so that he cannot see pictures on the pieces. Now he is looking for a solution of the puzzle. Normally the solution should exist but he is not sure whether Barborka has not replaced some pieces of the puzzle by pieces of another similar puzzle. Help him and write a program which reads a description of a set of puzzle pieces and decides whether it is possible to assembly the pieces into a rectangle with given side lengths or not.

Input

The input file consists of blocks of lines. Each block except the last describes one puzzle problem. In the first line of the block there are integers n and m, $0 < n, m \leq 6$ separated by one space. The integers n, m indicate the number of rows and columns in the puzzle respectively. The description of individual puzzle pieces is in the following $n \cdot m$ lines of the block. Each piece is a rectangle 3 centimeters wide and 4 centimeters high with possible juts or cavities in the middle of its sides. For each side of a puzzle piece just one of the following possibilities is true (see picture):

- there is no jut or cavity on the side, i.e., the side is flat – such sides can be used only on edges of the final picture when assembling the puzzle,
- there is one jut in the middle of the side,
- there is one cavity in the middle of the side.



As is usual, two pieces can be placed side by side only if one has a jut and the other has a cavity on corresponding sides. We will denote the flat sides by `F`, the sides with juts by `O` and the sides with cavities by `I`. Each piece is described by four letters characterizing its top, right, bottom, and left side. To make the task easier the pieces can be used only as they are described i.e. they cannot be turned.

After each block there is an empty line. The last block consists of just one line containing 0 0, i.e. two zeros separated by one space.

Output

The output file contains the lines corresponding to the blocks in the input file. A line contains `YES` if the corresponding block in the input file describes a puzzle that can be correctly assembled. Otherwise it contains `NO`. There is no line in the output file corresponding to the last "null" block of the input file.

Example

Input file:

```
3 5
FOOF
FOOI
FOOI
FOOI
FFOI
IOOF
IOOI
IOOI
IOOI
IFOI
IOFF
IOFI
IOFI
IOFI
IFFI

0 0
```

Output file:

```
YES
```

ACM International Collegiate Programming Contest 97/98

Sponsored by IBM

Central European Regional Contest

Problem G: Append

Input file: `append.in`
Output file: `append.out`
Program file: `append.pas` or `append.cpp`

Consider the following encoding scheme used in one famous compresion algorithm. Suppose we will code only sequences of lower case letters. Each such sequence of characters can be encoded to a sequence of pairs (p_i, r_i), where $p_i \geq 0$ is an integer and r_i is either a character (if $p_i = 0$) or an integer greater than zero and less or equal than p_i (if $p_i > 0$).

We describe now the decoding procedure for our encoding scheme. Let $(p_1, r_1), (p_2, r_2), \ldots$ be a code of a sequence. We get the sequence as follows: we take successively individual pairs of the code. If $p_i = 0$, then r_i is a character and we simply add r_i to the end of already decoded sequence. If $p_i > 0$, then r_i is an integer, $0 < r_i \leq p_i$, and we add to already decoded sequence r_i letters from this sequence starting at the position p_i places before the end.

For example, consider the sequence of pairs $(0, a), (1, 1), (0, b), (3, 3), (3, 3), (3, 2), (0, c)$. Decoding $(0, a)$ we get a. Decoding $(1, 1)$ we get aa. $(0, b)$ adds b getting aab. $(3, 3)$ will add aab, so now we have $aabaab$. Next pair $(3, 3)$ will again add aab, so we have $aabaabaab$. $(3, 2)$ will add aa, so our sequence is $aabaabaabaa$ and $(0, c)$ adds c. So the decoded sequence is $aabaabaabaac$. Note that in general for a given w it can exist more such sequences of pairs.

Let u, v be some sequences. By uv we will understand the sequence created by appending of the sequence v to the end of sequence u. Let C_w be a sequence of pairs which encodes a sequence of lowercase letters w. Suppose we have given a sequence of pairs C_w. The question is how many possibilities does exist for expressing the sequence C_w in the form $C_u C_v$ where u, v are sequences satisfying the equation $w = uv$ and neither u nor v is empty. Write a program that will answer this question.

Input

The input file consists of blocks of lines. Each block describes one sequence of pairs C_w to some w in such a way that the i-th line of the block contains either two integers p_i, r_i, $(r_i \leq p_i < 1000)$ separated by one space or 0 followed by one space and one character. Each block ends with one empty line.

Output

The output file contains the lines corresponding to the blocks in the input file. Each line contains the number of possibilities of representation of the sequence C_w in the form $C_u C_v$ where u, v are sequences satisfying the equation $w = uv$ and neither u nor v is empty.

Example

Input file:

```
0 a
1 1
0 b
3 3
3 3
3 2
0 c
```

⟨*one empty line*⟩

Output file:

```
1
```

ACM International Collegiate Programming Contest 97/98

Sponsored by IBM

Central European Regional Contest

Problem H: Gossiping

Input file: `gossip.in`
Output file: `gossip.out`
Program file: `gossip.pas` or `gossip.cpp`

There was a public bus transport in one town. All buses had circular lines. Each line had at least two stops. Some lines had some stops in common. When two or more bus drivers met on some stop they announced each other all news they knew, so that after leaving the stop they all knew the same news. All drivers started driving their buses at the same time and at this time each driver knew some news that was not known to any other driver. Each bus ran all the time along a fixed bus line. Different buses running along the same bus line started possibly on different stops of the bus line at the beginning.

The operation of buses was highly synchronized. The time necessary to get from one stop to next stop was equal for all stops and all lines.

There were n lines ($0 < n < 20$), d drivers (and also d buses) ($0 < d < 30$) numbered by integers from 1 to d, and s bus stops ($0 < s < 50$) numbered by integers from 1 to s.

The drivers' gossiping club would like to know whether each driver, in some time, would learn all news from his colleagues. Write a program that will answer this question.

Input

The input file consists of blocks of lines. Each block except the last describes one town. In the first line of the block there are integers n, d, and s described above separated by one space. The next $2n$ lines contain descriptions of n bus lines (2 lines for each bus line) as follows: In the first line there are stop numbers on the corresponding bus line separated by one space. The stops are listed in the order the bus passes them. After passing the last stop listed on the line the bus goes again to the first stop listed on the line. The second line describes on which stops the individual buses operating on the bus line started at the beginning. The description consists of pairs s_i, d_i, where s_i is a stop number on the bus line and d_i is the number of driver driving the bus. All numbers s_i, d_i on the line are separated by one space. The last block consists of just one line containing `0 0 0`, i.e. three zeros separated by one space.

Output

The output file contains the lines corresponding to the blocks in the input file. A line contains `Yes` if the corresponding block in the input file describes a situation where each driver will learn, in some time, all news from his colleagues. Otherwise it contains `No`. There is no line in the output file corresponding to the last "null" block of the input file.

Example

Input file:
```
2 3 5
1 2 3
1 1 2 2
2 3 4 5
2 3
0 0 0
```
Output file:
```
Yes
```