

ACM International Collegiate Programming Contest 2002–2003

Sponsored by IBM

Southwestern Europe Regional Contest

<http://swerc.up.pt/2002/>

Problem Set



University of Porto, Portugal

November 17th, 2002

This problem set should contain nine (9) problems on twenty-two (22) numbered pages. If something is missing from your problem set, please inform a runner immediately.

ACM International Collegiate Programming Contest 2002–2003

Southwestern European Regional Contest

University of Porto, Portugal
November 17th, 2002

Contents

Problem A: Solitaire	3
Problem B: Left labyrinths	5
Problem C: Crazy Search	7
Problem D: Intervals	9
Problem E: Family	11
Problem F: Timetable	13
Problem G: Word Puzzles	15
Problem H: Water Treatment Plants	19
Problem I: Servers	21

Remark

This problem set was developed jointly by the University of Porto and University of Warsaw for the SWERC'2002 and CERC'2002 regional contests.

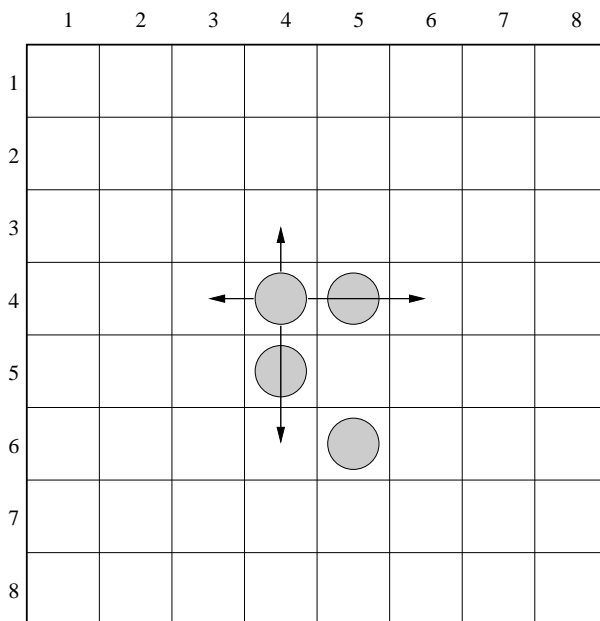
Problem A

Solitaire

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).



There are 4 moves allowed for each piece in the configuration shown 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.

Problem

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, \dots, 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 a configuration described in the second input line is reachable from the configuration described in the first input line in at most 8 moves, or one word NO otherwise.

Sample Input

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

Sample Output

```
YES
```

Problem B

Left labyrinths

“The instructions to turn always to the left reminded me that such was the common procedure for discovering the central courtyard of certain labyrinths.”

in *The garden of forking paths* by Jorge Luis Borges

Problem

A fellow librarian has exhumed in the Library of Babel a vast catalogue of labyrinths. It is our duty to classify them all and we count on your assistance. The plans of the labyrinths are already digitized and we need you to write a program to decide if the central courtyard of a given labyrinth can be reached following the simple procedure of always turning left at any intersection, and thus be called a *left labyrinth*.

The digitalization process reduces the plan of a labyrinth to a grid of cells, being each cell either a block of wall or just floor. Walls are sequences of contiguous blocks, forming either horizontal or vertical corridors between them. Each labyrinth has a single entrance, a hole in its exterior walls, and a single central courtyard. The courtyard differs from the corridors in the shape: a floor cell in a corridor has at least 2 block of wall on each side. You can assume that each plan has a single labyrinth and its outside walls can be contoured within the plan.

Input

The first two input lines are the integers, smaller than 100,

n - the number of lines in the map, and

m - the number of characters per line in the map.

The following n lines, each with m characters, have only two valid character values:

- (sharp) representing a block of **wall**;

. - (point) representing part of the **floor**.

Output

Your program must write either **YES** if the given labyrinth is a left labyrinth or **NO** otherwise

Problem C

Crazy Search

Problem

Many people like to solve hard puzzles some of which may lead them to madness. One such puzzle could be finding a hidden prime number in a given text. Such number could be the number of different substrings of a given size that exist in the text. As you soon will discover, you really need the help of a computer and a good algorithm to solve such a puzzle.

Your task is to write a program that given the size, N , of the substring, the number of different characters that may occur in the text, NC , and the text itself, determines the number of different substrings of size N that appear in the text.

As an example, consider $N=3$, $NC=4$ and the text "daababac". The different substrings of size 3 that can be found in this text are: "daa"; "aab"; "aba"; "bab"; "bac". Therefore, the answer should be 5.

Input

The first line of input consists of two numbers, N and NC , separated by exactly one space. This is followed by the text where the search takes place. You may assume that the maximum number of substrings formed by the possible set of characters does not exceed 16 Millions.

Output

The program should output just an integer corresponding to the number of different substrings of size N found in the given text.

Sample Input

```
3 4
daababac
```

Sample Output

```
5
```


Problem D

Intervals

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

Problem

Write a program that:

- reads the number of intervals, their endpoints and integers c_1, \dots, 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 $i = 1, 2, \dots, n$,
- 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 $i + 1$ -th line of the input contains three integers a_i , b_i and 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 set Z sharing at least c_i elements with interval $[a_i, b_i]$, for each $i = 1, 2, \dots, n$.

Sample Input

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

Sample Output

```
6
```


Problem E

Family

We want to find out how much related are the members of a family of monsters. 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. This is impossible, however, since the number of genes is very large. Still, we do know the family tree (well, not actually a tree, but you cannot really blame them, these 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. For if A and B had common genes, these would be necessarily inherited by both C and D .

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

Problem

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 ($2 \leq n \leq 300$) is the number of members in a family. Family members are numbered arbitrarily from 1 to n . Integer k ($0 \leq k \leq n - 2$) 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 ($1 \leq m \leq n^2$) — 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 numbers 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 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.

Sample Input

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

Sample Output

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

Problem F

Timetable

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), not stopping anywhere between. On each station a departure timetable is available. 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 however — 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 same day.

Problem

Write a program that:

- reads the number n and departure timetable for each of n cities 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 $1, 2, \dots, n$ 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 than one optimal connection with the same departure and arrival time, your program should output just one.

Sample 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
```

Sample Output

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

Problem G

Word Puzzles

Word puzzles are usually simple and very entertaining for all ages. They are so entertaining that Pizza-Hut company started using table covers with word puzzles printed on them, possibly with the intent to minimise their client's perception of any possible delay in bringing them their order.

Even though word puzzles may be entertaining to solve by hand, they may become boring when they get very large. Computers do not yet get bored in solving tasks, therefore we thought you could devise a program to speedup (hopefully!) solution finding in such puzzles.

The following figure illustrates the PizzaHut puzzle. The names of the pizzas to be found in the puzzle are: MARGARITA, ALEMA, BARBECUE, TROPICAL, SUPREMA, LOUISIANA, CHEESEHAM, EUROPA, HAVAIANA, CAMPONESA.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	Q	W	S	P	I	L	A	A	T	I	R	A	G	R	A	M	Y	K	E	I
1	A	G	T	R	C	L	Q	A	X	L	P	O	I	J	L	F	V	B	U	Q
2	T	Q	T	K	A	Z	X	V	M	R	W	A	L	E	M	A	P	K	C	W
3	L	I	E	A	C	N	K	A	Z	X	K	P	O	T	P	I	Z	C	E	O
4	F	G	K	L	S	T	C	B	T	R	O	P	I	C	A	L	B	L	B	C
5	J	E	W	H	J	E	E	W	S	M	L	P	O	E	K	O	R	O	R	A
6	L	U	P	Q	W	R	N	J	O	A	A	G	J	K	M	U	S	J	A	E
7	K	R	Q	E	I	O	L	O	A	O	Q	P	R	T	V	I	L	C	B	Z
8	Q	O	P	U	C	A	J	S	P	P	O	U	T	M	T	S	L	P	S	F
9	L	P	O	U	Y	T	R	F	G	M	M	L	K	I	U	I	S	X	S	W
10	W	A	H	C	P	O	I	Y	T	G	A	K	L	M	N	A	H	B	V	A
11	E	I	A	K	H	P	L	B	G	S	M	C	L	O	G	N	G	J	M	L
12	L	D	T	I	K	E	N	V	C	S	W	Q	A	Z	U	A	O	E	A	L
13	H	O	P	L	P	G	E	J	K	M	N	U	T	I	I	O	R	M	N	C
14	L	O	I	U	F	T	G	S	Q	A	C	A	X	M	O	P	B	E	I	O
15	Q	O	A	S	D	H	O	P	E	P	N	B	U	Y	U	Y	O	B	X	B
16	I	O	N	I	A	E	L	O	J	H	S	W	A	S	M	O	U	T	R	K
17	H	P	O	I	Y	T	J	P	L	N	A	Q	W	D	R	I	B	I	T	G
18	L	P	O	I	N	U	Y	M	R	T	E	M	P	T	M	L	M	N	B	O
19	P	A	F	C	O	P	L	H	A	V	A	I	A	N	A	L	B	P	F	S

Problem

Your task is to produce a program that given the word puzzle and words to be found in the puzzle, determines, for each word, the position of the first letter and its orientation in the puzzle.

You can assume that the left upper corner of the puzzle is the origin, (0,0). Furthermore, the orientation of the word is marked clockwise starting with letter A for north (note: there are 8 possible directions in total).

Input

The first line of input consists of three positive numbers, the number of lines, $0 < L \leq 1000$, the number of columns, $0 < C \leq 1000$, and the number of words to be found, $0 < W \leq 1000$. The following L input lines, each one of size C characters, contain the word puzzle. Then at last the W words are input one per line.

Output

Your program should output, for each word (using the same order as the words were input) a triplet defining the coordinates, line and column, where the first letter of the word appears, followed by a letter indicating the orientation of the word according to the rules define above. Each value in the triplet must be separated by one space only.

Sample Input

```
20 20 10
QWSPILAATIRAGRAMYKEI
AGTRCLQAXLPOIJLFBVBUQ
TQTKAZXVMRWALEMAPKCW
LIEACNKAZXKPOTPIZCEO
FGKLSTCBTROPICALBLBC
JEWHJEEWSMLPOEKORORA
LUPQWRNJOAAGJKMUSJAE
KRQEIOLOAOQPRTVILCBZ
QOPUCAJSPPOUTMTSLPSF
LPOUYTRFGMMLKIUISXSW
WAHCPOIYTGAKLMNAHBVA
EIAKHPLBGSMCLOGNGJML
LDTIKENVCSWQAZUAOEAL
HOPLPGEJKMNUTIIORMNC
LOIUFTGSQACAXMOPBEIO
QOASDHOPEPNBUYUYOBXB
IONIAELOJHSWASMOUTRK
HPOIYTJPLNAQWDRIBITG
LPOINUVMRTEMPTMLMNBO
PAFCOPLHAVAIANALBPFS
MARGARITA
ALEMA
BARBECUE
TROPICAL
SUPREMA
LOUISIANA
CHEESEHAM
EUROPA
HAVAIANA
CAMPONESA
```

Sample Output

```
0 15 G
2 11 C
7 18 A
4 8 C
16 13 B
4 15 E
10 3 D
5 1 E
19 7 C
11 11 H
```

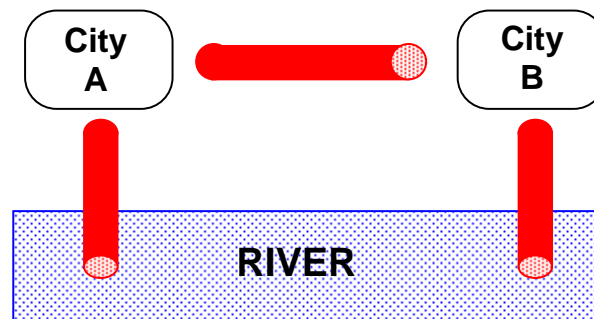

Problem H

Water Treatment Plants

River pollution control is a major challenge that authorities face in order to ensure future clean water supply. Sewage treatment plants are used to clean-up the dirty water coming from cities before being discharged into the river.

As part of a coordinated plan, a pipeline is setup in order to connect cities to the sewage treatment plants distributed along the river. It is more efficient to have treatment plants running at maximum capacity and less-used ones switched off for a period. So, each city has its own treatment plant by the river and also a pipe to its neighbouring city upstream and a pipe to the next city downstream along the riverside. At each city's treatment plant there are three choices:

- **either** process any water it may receive from one neighbouring city, together with its own dirty water, discharging the cleaned-up water into the river;
- **or** send its own dirty water, plus any from its downstream neighbour, along to the upstream neighbouring city's treatment plant (provided that city is not already using the pipe to send its dirty water downstream);
- **or** send its own dirty water, plus any from the upstream neighbour, to the downstream neighbouring city's plant, if the pipe is not being used.



The choices above ensure that:

- every city must have its water treated somewhere and
- at least one city must discharge the cleaned water into the river.

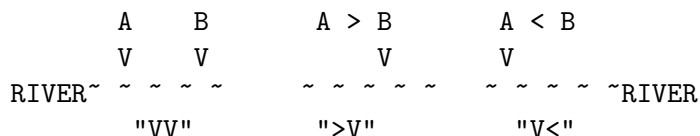
Let's represent a city discharging water into the river as "V" (a downwards flow), passing water onto its neighbours as ">" (to the next city on its right) or else "<" (to the left). When we have several cities along the river bank, we assign a symbol to each (V, < or >) and list the cities symbols in order. For example, **two** cities, A and B, can

- each treat their own sewage and each discharges clean water into the river. So A's action is denoted V as is B's and we write "VV" ;

- or else city A can send its sewage along the pipe (to the right) to B for treatment and discharge, denoted ">V";
- or else city B can send its sewage to (the left to) A, which treats it with its own dirty water and discharges (V) the cleaned water into the river. So A discharges (V) and B passes water to the left (<), and we denote this situation as "V<".

We could not have "><" since this means A sends its water to B and B sends its own to A, so both are using the same pipe and this is not allowed. Similarly "<<" is not possible since A's "<" means it sends its water to a non-existent city on its left.

So we have just **3** possible set-ups that fit the conditions:



If we now consider **three** cities, we can determine **8** possible set-ups.

Problem

Your task is to produce a program that given the number of cities *NC* (or treatment plants) in the river bank, determines the number of possible set-ups, *NS*, that can be made according to the rules define above.

You need to be careful with your design as the number of cities can be as large as 100.

Input

The input consists of a sequence of values, one per line, where each value represents the number of cities.

Output

Your output should be a sequence of values, one per line, where each value represents the number of possible set-ups for the corresponding number of cities read in the same input line.

Sample Input

2
3
20

Sample Output

3
8
102334155

Problem I

Servers

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 with 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 (transmission time-wise) 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 to 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 $30n$.

Problem

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

In the first line there are 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.

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

In the following m lines the wires are described. 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.

Sample Input

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

Sample Output

9

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