Problem A Word Finder 10 points

Word finder puzzles are a popular amusement. In such a puzzle you are given a rectangular grid of letters, and a list of words which you are asked to locate. These words may begin anywhere in the grid and proceed vertically, horizontally, or diagonally, and either forwards or backwards.

Write a program that will read in details of such a puzzle and report the whereabouts of each word in the list or that it cannot be found anywhere in the grid.

Given a sequence of such grids, and a list of target words for each, the problem is to find the occurrence of each target word in the grid, specifying the start and end positions, as (row, column) pairs, counting rows from 1 downwards, and columns from 1 leftwards. Each target word will appear at most once in the grid.

Input will consist of a series of puzzles. Each puzzle will start with a line containing a pair of integers representing the number of rows and number of columns in the grid (*r* and *c*, both in the range 1 to 20 inclusive). This is followed by r rows of the grid, each containing c uppercase letters. This is followed in turn by a series of lines, each containing between 3 and the maximum of r and c uppercase letters, representing the target words, one per line,each consisting of a string of uppercase characters. This list is terminated by a line containing a single #. Input is terminated by a line containing two zeroes (0 0).

For each grid output a header line consisting of the word “Grid” followed by a single space, and a running number starting from 1. This is followed by the list of target words in the order they appear in the input, each on a line by itself, followed by a space and either the words “not found” if the word was not found in the grid or the locations of the first and last characters in the word, formatted exactly as shown below. Separate successive grids by single blank lines.

Sample Input

```
3 4
EGAR
TOOL
ATSX
RAGE
FROG
GOT
ROT
#
1 3
ABC
XYZ
#
0 0
```

Sample Output

```
Grid 1
RAGE found from (1,4) to (1,1).
FROG not found.
GOT found from (1,2) to (3,2).
ROT found from (1,4) to (3,2).

Grid 2
XYZ not found.
```

Problem B Join the Dots 10 points

Many children's books contain simple pictures from which all or most of the lines have been removed. Before removal of the lines, the vertices (corners) of the lines are numbered so that if they are joined in order the original picture is reproduced.

Write a program that will perform this task. Given the difficulties of specifying a single point with an integer which could contain several digits, the picture will be represented as a series of coordinates in the order in which they need to be connected. According to the rules to be given later, use one of '–', '|', '/' and '\' for simple lines, use '+ if two lines cross or meet and use '*' if more than two lines cross and/or meet. Given that with these symbols you can only draw straight lines parallel to or at 45° to the axes, draw a diagonal line first for as long as necessary, followed by a vertical or horizontal line, where 'first' means as you move from the current point to the next point

Input will be a series of scenarios. Each scenario will start with the size of the picture (the number of rows, *r*, and the number of columns, *c*, both in the range 1 to 20 inclusive).This line will be followed by a series of lines specifying the position of the points, terminated by a line containing two zeroes (0 0). You can assume that all points will lie within the range specified for that picture and that no pointwill be the same as its immediate predecessor. Input will be terminated by a null picture, i.e. one with *r* and *c* both zero (0 0). This 'picture' should not be processed.

Output will consist of a series of pictures, one for each scenario in the input. Each picture will consist of *r* lines of text each containing *c* characters, where each character is either a space or one of those specified above. Note that terminal spaces **must** be written. Successive pictures must be separated by a blank line. Follow the examples below, where spaces have been replaced by 'Δ' for clarity.

Sample Input:

```
6 12
1 6
1 12
6 9
0 0
0 0
```

Sample Output

```
ΔΔΔΔ------+
ΔΔΔΔΔΔΔΔΔ/Δ
ΔΔΔΔΔΔΔΔ/ΔΔ
ΔΔΔΔΔΔΔ/ΔΔΔ
ΔΔΔΔΔΔΔ|ΔΔΔ
ΔΔΔΔΔΔΔ|ΔΔΔ
```

<<It would be nice to have another example, preferably with a multiple crossing. Richard?, Stewart?>>

This is the same output with . instead of Δ for the spaces. Comments please (note this will only show in the printable versions)

```
.....------+
.........../.
.........../..
.........|...
.........|...
```

Problem C Colour by Numbers 10 points

Most children at some stage are given a kit with a picture, consisting of a series of numbered regions, and a set of paints numbered in the same way. The child is expected to apply the correctly numbered paint to each numbered region to produce a, typically garishly, coloured picture.

Write a program to perform a similar task, given a series of ‘pictures’ and ‘paints’. The pictures will be represented as grids, the paints as symbols. The grids will be represented as rectangles of symbols, surrounded by a border. The grid will consist of characters chosen from the set {space, ‘|’, ‘–’, ‘\’, ‘/’}, where the non-space characters will divide the grid into regions that otherwise contain spaces. All lines of non-space characters will terminate either at the edge of the picture or at another line. The regions are effectively numbered by the order of appearance of their first characters in a left to right, top to bottom scan.

Input will be a series of pictures. Each picture will start with a line specifying its size as the number of rows and the number of columns (*r* and *c*, with both in the range 1 to 20, both inclusive). This will be followed by *r* rows, each containing *c* characters, chosen from the above set. You can assume that the outermost lines of characters will be non-space characters. This will be followed by a line of characters, excluding those in the above set. These characters are to be applied to the regions in the picture, i.e. fill the first region with the first character, the second region with the second character, and so on. There will be at least enough characters to fill all the regions, but you cannot assume that they will all be different. Input will be terminated by a line containing two zeroes (0 0), specifying an empty picture. This should not be processed.

Output will be a series of pictures, one for each specified in the input. Each picture will consist of a grid of characters the same size as specified in the input, with each blank region replaced by the relevant character. Leave one blank line between pictures.

Sample Input

```
5 7
-------
|  |  |
|---  |
|     |
-------
%##@
5 7
-------
|  |  |
|  ---|
|  |  |
-------
%##@
0 0
```

Sample Output:

```
-------
|%%|##|
|---##|
|#####|
-------

-------
|%%|##|
|%%---|
|%%|##|
-------
```

Problem D Muggle Money 10 points

Those of you who know the Harry Potter books will know that there are two interpenetrating populations of humans on this planet — those who can do magic (usually known as wizards) and those who can't (known as Muggles). Because the race of wizards was dying out they interbred with the Muggles, so there are now purebred wizards, purebred Muggles and wizard-Muggle crosses, known disparagingly as Mudbloods. Anyone with wizard blood can do magic and therefore has to attend wizard school (Hogwarts Academy) to learn how. In order to buy their school supplies — robes, wand, cauldron, books, etc —from the wizard shops, they obviously need to have access to wizard money. This is not a problem for those living in the wizard world, but it is for those in the Muggle world, and this is where you come in.

Wizard money consists of brass knuds, silver sickles, and gold galleons. There are 29 knuds to a sickle and 17 sickles to a galleon. Muggle money consists of pounds and pence, with 100 pence to a pound. Write a program that, given an exchange rate, and an amount of money in one system, will convert it to the other system.

Input will consist of a series of conversions. Each conversion consists of a single character identifier followed by an exchange rate (a real number given with up to 4 decimal places) followed by two or three integers. The character is either 'M' for Muggle money in pounds and pence to be converted to wizard money or 'W' for wizard money in galleons, sickles and knuds to be converted into pounds and pence. The rate is always the amount of the largest unit of the desired currency that can be bought by one of the largest unit of the given currency. Thus the line starting "M 0.9842" means that we are converting Muggle money to wizard money and that 1 pound will buy 0.9842 galleons. Input will be terminated by a line consisting of only a '#'. This line should not be processed.

Output will consist of one line for each line of input. It will consist of the amount of money in the given units, the words 'converts to' and the converted amount, again identified. Knuds and pence are to be rounded to the nearest integer, with 0.5 being rounded up. Note that 'galleon', 'sickle', 'knud', 'pound' and 'penny' are the singular forms and 'galleons', 'sickles', 'knuds', 'pounds' and 'pence' are the plural forms. Follow the example given.

Sample input

```
W 1.00 1 1 1
M 1.00 1 1
M 0.9842 20 37
#
```

Sample output

```
1 galleon 1 sickle 1 knud converts to 1 pound 6 pence.
1 pound 1 penny converts to 1 galleon 0 sickles 5 knuds.
20 pounds 37 pence converts to 20 galleons 0 sickles 24 knuds.
```

Problem E Round Robin 30 points

In many games and sports there are tournaments where several contestants (individuals or teams) enter a Round Robin contest, i.e. each contestant plays every other contestant. The final ranking is then typically determined by the number of wins usually followed by some way of breaking ties.

In Tournament Scrabble™ there are *n* players in a grade playing *n* -1 games each for a total of *n*(*n* -1)/2 games. For each game the players will submit a summary specifying their names, final scores and the number of bonus words each got. From these one can determine the winner (the one with the higher score) and the spread (the difference between the scores of the two players, counted as positive for the winner and negative for the loser). At the end of the tournament players are ranked by as many of the following criteria (in order) as necessary to resolve ties: wins (a draw is considered as half a win each), total spread, average score, and number of bonus words. If there is still a tie the players are listed lexicographically, i.e. according to the ordering specified by the ASCII representations of their names).

Write a program that will read the game records for a Round Robin and produce a table ranking the players according to the above rules.

Input will consist of a series of tournaments. The first line of each tournament will contain the number of players in the tournament (*n*), where *n* is even and in the range 4 to 24, both numbers inclusive. This will be followed by *n* lines containing the names of the players as strings of up to 10 characters, without leading or embedded spaces, followed in turn by *n*(*n* -1)/2 lines containing game summaries for the individual games as given above and shown in the sample input below. You cannot assume that this will necessarily be in 'rounds', i.e. that all *n* players will appear in each block of *n*/2 games, however you can assume that each player will appear *n*–1 times (once with each other player). Each game summary line will consist of two records separated by a space where each record contains a name, the score for that person, and the number of bonus words, all separated by spaces. The file will be terminated by a line containing a single zero (0).

Output will consist a series of rankings, one for each tournament specified in the input. Each ranking starts with a line containing the words “Tournament number”, followed by a blank and the tournament number (a running number starting at 1, followed by a listing of the *n* players in the above order, one per line. Each line will consist of the name of the player, left justified in a field of width 10, the total number of wins plus floor(number of draws divided by 2) right justified in a field of width 3, a ‘+’ if the number of draws is odd otherwise a space, the spread right justified in a field of width 6, the average score rounded to the nearest integer (0.5 rounded up) right justified in a field of width 4 and the number of bonus words right justified in a field of width 3. Separate successive tournaments by a blank line.

Sample Input

```
4
Player1
Player2
Player3
Player4
Player1 412 2 Player2 330 0
Player3 309 0 Player4 384 1
Player1 445 2 Player3 394 1
Player2 414 3 Player4 317 0
Player1 395 1 Player4 271 0
Player2 463 2 Player3 312 0
0
```

Sample Output

```
Tournament number 1
Player1     3    257 417  5
Player2     2    166 402  5
Player4     1   -146 324  1
```

```
Player3     0   -277 338  1
```

Problem F Morse Code 30 points

In international Morse code, sequences of dots (.) and dashes (–) are used to represent letters as follows.

```
A .-      N -.
B -...    O ---
C -.-.    P .--.
D -..     Q --.-
E .       R .-.
F ..-.    S ...
G --.     T -
H ....    U ..-
I ..      V ...-
J .---    W .--
K -.-     X -..-
L .-..    Y -.--
M --      Z --..
```

In telegraphic communication the dots and dashes are distinguished from one another by their length, and spacings between them are used to demarcate individual letters. Without these spaces even short pieces of code are highly ambiguous. For example: .--- could be any one of: ETTT, ETM, EMT, EO, ATT, AM, WT, or J.

Write a program to produce all possible translations of such a sequence of dots and dashes into letters.

Input will be a sequence of lines containing code sequences (groups) written as dots and dashes without intervening, leading or trailing spaces. Each group will contain between 1 and 10 characters. Input is terminated by a line containing a single #.

For each line in the input, output a line containing the word “GROUP”, a space and the dot-dash group, followed by all possible translations of that group, each on a separate line. All the translations where the first character is replaced by a letter come first, then all those where the first two are replaced by a letter, and so on, and this same ordering is also present within each block. Successive groups should be separated by a single blank line.

Sample Input

```
.--
-----
#
```

Sample Output

```
GROUP .--
ETT
EM
AT
W

GROUP -----
TTTTT
TTTM
TTMT
TTO
TMTT
TMM
TOT
MTTT
MTM
MMT
MO
OTT
OM
```

Problem G No Minus Signs 30 points

It is well known that the decimal (base 10) numbers that humans use are represented in the computer using binary (base 2). For example, the decimal number 25 (= 2 • 10^1 + 5 • 10^0) is stored as the binary number 11001 = 1 • 2^4 + 1 • 2^3 + 0 • 2^2 + 0 • 2^1 + 1 • 2^0.

The usual internal representation for negative integers is 2's complement, which uses one bit to represent the sign. Similarly, decimal numbers are prefixed by an extra character ('-' or ' ' as appropriate). The management of Crackpot Computers thinks it can save money by representing both positive and negative integers using only digits {0, 1, 2, … }, without the need for negative signs, by using base $-b$ representations. For example, the decimal number –25 is represented as 35 in base –10 since $35_{-10} = 3 \cdot (-10)^1 + 5 \cdot (-10)^0 = -30 + 5 = -25$.

Likewise, for base –2 we have

$$\begin{aligned} 111011_{-2} &= 1 \cdot (-2)^5 + 1 \cdot (-2)^4 + 1 \cdot (-2)^3 + 0 \cdot (-2)^2 + 1 \cdot (-2)^1 + 1 \cdot (-2)^0 \\ &= -32 + 16 - 8 - 2 + 1 \\ &= -25 \\ &= -11001_2 \end{aligned}$$

To be compatible with other systems your job as an employee of Crackpot is to write a program that will convert integers (both positive and negative) from a given positive base representation to the corresponding negative base representation. Input will be a sequence of integer n_b and base b pairs, where n_b is at most 8 digits long and b is in the range $2 \le b \le 10$. The required output is a number written in base $-b$ for each input line. Input will be terminated by the pair 0 0, and this line should not be processed.

Sample Input

```
–25 10
–92 10
–181 10
–10101 2
75 8
0 0
```

Sample Output

```
35
1908
1999
111111
115
```

Problem H Holidays 30 points

Rules for determining when statutory holidays are to be *observed* are complex. Employers need to know these dates to comply with employment law. In most countries there are holidays of the following types:

F Fixed — the holiday occurs on the same date every year.

W Weekday — the holiday occurs on some particular date, unless that day is a weekend in which case observance of the holiday is moved to the next working day (which is usually a Monday).

N The n'th weekday in a certain month. For example, the second Friday in November.

It is possible for conflicts to occur. If, for example, Cockroach Day fell on the 4th Monday in December then in some years Christmas Day and Cockroach Day would fall on the same day. It would be unfair for workers to miss a holiday in such situations, so the observance of one of the holidays should be moved to the next working day. A similar situation occurs if an W-type holiday occurs during a weekend.

Observances of holidays for any particular year are to be allocated in the following way. All type F holidays are allocated first, then all type W and finally all type N. For each type X:

1. Holidays of type X are worked through in increasing date order. For each holiday, if X = :
   F: if the day is free, allocate it, otherwise there is a conflict.
   W: if the day is free and it is a weekday, allocate it, otherwise there is a conflict.
   N: if the day is free, allocate it, otherwise there is a conflict.

2. Conflicting holidays of type X are worked through in increasing date order. For each holiday, if X = :
   F: choose the next free day
   W: choose the next free weekday
   N: choose the next free occurrence of that day (e.g. for a Thursday holiday, the next free Thursday).

Notes

1) Years divisible by 4 are leap years, years divisible by 100 are not leap years and years divisible by 400 are leap years, thus 1976 and 2000 are leap years; 1900 and 1953 are not.
2) January 1st, 1900 was a Monday.
3) Any holidays that get pushed beyond the end of December are lost.

Write a program to determine when holidays are to be observed, according to the preceding rules.

Input takes the form of a number of scenarios. Each scenario consists of the name of the scenario followed by a calendar. The input is terminated by a 'scenario name' consisting of #. Each calendar consists of a number of holiday specifications (in no particular order), followed by a list of years for which holiday calendars are required. Each holiday specification consists of a single character representing the type of holiday (W, F or N), two or three integers specifying when the holiday occurs, a space, and the name of the holiday (a string of up to 40 characters). The integers are:

- for types W and F, the date in the form month day. Note that February 29th (2 29) is interpreted as March 1st in non-leap years.
- for type N, the month, the day number (1 = Monday, 7 = Sunday) and an 'ordinal' in the range 1 to 5, thus "N 11 5 2" means the second Friday in November. Note that specification of the 5th occurrence of a day may imply the first occurrence of that day in the next month.

The end of the list is marked by a line containing only a # (as the 'type' of the holiday).

The list of years for which holiday calendars is required is a list of integers in the range 1900 – 2100 (both years inclusive) on one or more lines. The end of this list is marked by a single 0.

Output consists of a number of holiday calendars, one for each specified year for each scenario. Each holiday calendar start with a line containing the name of the scenario, followed by a space and the words "holidays for" followed by a space and the year. This is followed by a list of holiday days each of which consists of the date on which it will be observed (the list is sorted by this field), the date on it would have been observed in the absence of conflicts, and the holiday name, with one space between each field. Follow the example below. Each holiday calendar is separated from the following one by a blank line.

Sample input

```
New Zealand
W 1 1 New Year's Day
W 1 2 Day after new year's day
F 2 6 Waitangi Day
W 4 23 Otago Day
F 4 25 ANZAC day
N 6 1 1 Queen's birthday
N 10 1 4 Labour day
N 11 5 2 Christchurch show day
W 12 25 Christmas Day
W 12 26 Boxing Day
#
1999
2000
0
```

Sample output

```
New Zealand holidays for 1999
1999-01-01 1999-01-01 New Year's Day
1999-01-04 1999-01-02 Day after new year's day
1999-02-06 1999-02-06 Waitangi Day
1999-04-23 1999-04-23 Otago Day
1999-04-25 1999-04-25 ANZAC day
1999-06-07 1999-06-07 Queen's birthday
1999-10-25 1999-10-25 Labour day
1999-11-12 1999-11-12 Christchurch show day
1999-12-27 1999-12-25 Christmas Day
1999-12-28 1999-12-26 Boxing Day

New Zealand holidays for 2000
2000-01-03 2000-01-01 New Year's Day
2000-01-04 2000-01-02 Day after new year's day
2000-02-06 2000-02-06 Waitangi Day
2000-04-24 2000-04-23 Otago Day
2000-04-25 2000-04-25 ANZAC day
2000-06-05 2000-06-05 Queen's birthday
2000-10-23 2000-10-23 Labour day
2000-11-10 2000-11-10 Christchurch show day
2000-12-25 2000-12-25 Christmas Day
2000-12-26 2000-12-26 Boxing Day
```

Problem I Three Coins in a Fountain 100 points

Mark has three coins. One is two-headed (2H), one is standard (S), and one is two-tailed (2T). Actually the 2H and 2T coins are standard as well in the sense that their sides can be told apart. For the purposes of maximal confusion these sides will be called heads and tails as well.

Mark produces a sequence of H's and T's as follows. He chooses any of the three coins, flips it, and records the result (always H for 2H, the actual result for S, and always T for 2T). Based on the actual (not recorded) result of the toss he may choose another coin and proceed in the same fashion. The rules for his choices are as follows:

Original coin	True Result	New Coin
2H	H	2H
2H	T	S
S	H	2H
S	T	2T
2T	H	S
2T	T	2T

(Basically, if he is using a non-standard coin he keeps it if the result agrees with its type, and switches to standard if it doesn't. Using the standard coin he always switches to the unfair coin of the same type as the actual result.)

Write a program to determine in how many different possible ways a given sequence of recorded results could have arisen. For example the sequence HH can be obtained in five ways

```
S(H),  2H(H)
S(H),  2H(T)
2H(H), 2H(H)
2H(H), 2H(T)
2H(T), S(H)
```

where for each toss we have shown the coin type tossed, then the actual result in parentheses. As we see above, two sequences of tosses are considered different if there is any toss in which they use different coins, or use the same coin but with differing results.

Input will consist of a sequence of lines each consisting of alternating characters (H or T) and integers (greater than zero). For example:

```
H 3 T 17 H 4 H 2
```

which indicates a sequence of 3 heads, followed by 17 tails, then 4 heads, then 2 more heads (note we do not guarantee alternating heads and tails in the sequence). There will never be more than 40 flips, i.e. the sum of all the integers will be less than or equal to 40. The end of input will be specified by a line containing only the character #.

For each line in the input output the number of ways in which that sequence could have arisen.

Sample Input

```
H 2
H 3 T 17 H 4 H 2
H 40
#
```

Sample Output

```
5
38493
433494437
```

Problem J Forced Games 100 points

A typical two-player game with a finite number of configurations (e.g. chess pieces positioned on a board, noughts and crosses written on a grid) can be represented as a bipartite graph, a special type of directed graph. Let us call the players P1 and P2. Each vertex in the graph represents a board configuration. Some vertices (set V1) are board configurations in which P1 has the next turn, while other board configurations (set V2) are board configurations in which P2 has the next turn. Each edge must travel from a vertex in V1 to a vertex in V2, or *vice versa* (this is the "bipartite" property).

Some vertices will correspond to a win for the player who has just moved (placed a chess piece, or written a nought or a cross). A common task for programs that play games of this type is to determine whether it is possible for a player to force a win from the current position. For player P1 (P2), let us call this set of vertices W1 (W2). A vertex v is in W1 (W2) if:

- v is specified as a winning position for P1 (P2).
- v is in V1 (V2) and there exists at least one move from it to a vertex in W1 (W2).
- v is in V2 (V1) and **all** moves from it are to vertices in W1 (W2).

Write a program that will read a graph specifying game configurations and moves and produce W1 and W2.

Input will consist of a series of game (connected graph) descriptions, separated from each other by blank lines. Each description will consist of an integer n ($2 \le n \le 100$) followed by n lines of adjacency lists for vertices 0 to n–1. Each vertex either represents a game state in which one or more valid moves exist (in which case the adjacency list is a list of all vertices that can be moved to) or represents the end of a game (in which case one of the following integers appears: –1 if the vertex is in W1, –2 if the vertex is in W2, or –3 if the vertex represents a draw). At least one vertex will have an adjacency list of -1 and at least one vertex will have an adjacency list of –2. Input will be terminated by a value of 0 for n.<<Do we need a note here stating what this implies in terms of membership of the various sets?>>

The output format for each game depends on whether the input graph is valid. If the graph is not bipartite then a line containing "Not bipartite" should be printed. Otherwise output consists of the sets W1 and W2, on separate lines, each specified as a list of integers separated by single spaces and sorted into ascending order. A blank line is used to separate output from successive games.

Sample Input

```
3
1 2
-1
-2

12
-1
2 3
-1
8
5
-3
4 7
8
-2
10 0
5
1 6 9

0
```

Sample Output

```
Not bipartite

0 1 2 9
3 7 8
```

Problem K Pizza Wars 100 points

The city of Kroy Wen is laid out in a perfect rectangular grid. Unfortunately, despite this, its roads were laid out in a somewhat haphazard fashion. A city map shows four different types of location:

. An apartment building
- A street running EW
| A street running NS
+ An intersection

Part of the map might look as follows (numbers and letters for later reference only)

```
    A   B   C   D   E
1   .   |   -   +   .
2   .   |   |   |   .
3   +   -   +   +   .
```

The rules of the road are that from a NS street (|) you can enter the grid square immediately above or below if that square contains either another NS street or an intersection (but not an EW street). Similar rules apply to EW streets. From an intersection you can access intersections or EW streets to the left and right, and intersections or NS streets above or below.

In the example grid above, you can drive from square C1 to square C2 but only via D1, D2, D3, and C3. You can drive from B1 to B2 and back again, but cannot connect from these two squares to the rest of the road network.

Situated around Kroy Wen are a number of pizza restaurants who are in heated competition for business from the various apartment buildings. The residents of Kroy Wen are very time conscious, and place speed of delivery above almost all other criteria for choosing which company they order their pizza from. This speed is determined solely by the distance from the restaurant via the road network to a square adjacent (horizontally or vertically on the map) to an apartment.

Suppose that in the example above there were three restaurants P2G at C3, PHT at B1, and EBP at D2. For the five apartments we check the distances to each of the three restaurants:

```
        P2G     PHT     EBP
A1      *       0       *
A2      2       1       4
E1      3       *       1
E2      2       *       0
E3      1       *       1
```

(*'s denote inaccessible apartments for that restaurant).

The rules for how an apartment's pizza business get allocated are as follows: If there is a "clear closest" restaurant it gets all the business. Here clear closest means that all other restaurants are at least two blocks further away. If there are one or more closest restaurants and zero or more "near misses" (one block further away) then each closest restaurant gets 3 points, and the near misses 1 point each. Business from that apartment is allocated according to the fraction of the total number of points for that apartment which each business receives. Again in the example above (counting an apartment as 1 unit)

```
        P2G     PHT     EBP
A1      0       1       0
A2      0.25    0.75    0       (PHT receives 3 points, P2G 1)
E1      0       0       1
E2      0       0       1
E3      0.5     0       0.5     (PHT and EBP each receive 3 points)

Total   0.75    1.75    2.5
```

Write a program that will determine the total amount of business that each restaurant will get, given a map, and the locations of the pizza restaurants.

Input will consist of a series of scenarios (maps). Each map will start with a line containing a pair of integers giving the number of rows and columns in the map (r, c, $3 \le r\ c \le 100$), followed by r lines each containing c characters from the set {“x”, “+”, “–”, ‘|”}. This will be followed by between two and twenty lines containing the name of each restaurant (a string of 3 characters) and a pair of integers, representing the row number and column number in which it is located. The left column and top row are both numbered 1). This list will be terminated by a line containing only a single character #. Note that zoning laws preclude pizza restaurants from sharing a location and also from occupying the same location as an apartment block.) Input will be terminated by a line containing two zeroes (0 0) for the size of the map.

Output will consist of a set of allocations, one set for each map in the input. The first line will contain the words “Map number” followed by a space and a running number starting at 1. This will be followed by a series of lines, one for each restaurant, specifying the name of the restaurant and their share of the trade in the form shown below with all numbers rounded to two decimal places. Leave a blank line between successive maps.

Sample Input

```
3 5
.|-+.
.|||.
+-++.
P2G 3 3
PHT 1 2
EBP 2 4
#
0 0
```

Sample Output

```
Map number 1
P2G will deliver to 0.75 apartments.
PHT will deliver to 1.75 apartments.
EBP will deliver to 2.50 apartments.
```

<<We need to figure out and specify the order in which restaurants are to be output — as given, alphabetical, rank order?>>

Problem L Fuzzy Fences 100 points

In the land of Fuzz one's status is determined by the length of one's fence. These fences are ordinary post and rail fences with a single rail connecting adjacent posts. The posts are standard but the lengths of the rails are determined by the local shaman. The length of the fence (up to 5 kilometers) is equal to the sum of the lengths of the constituent rails.

When a fence is to be built the shaman will consult the spirits — actually the local lumber merchant (who just happens to be his brother-in-law) to determine the most profitable (oops, propitious) lengths of rails. He then specifies a set of possible positions for the posts, typically far more than is necessary. There is presumably some nepotism involved in this, although this has not been demonstrated, because the set of positions can always be connected by rails of the specified lengths.

Given all this, the land owner wants to build the fence as cheaply as possible. The lumber merchant will quote a price for each of the required lengths of rail and can produce as many rails as necessary. There is no obvious relationship between price and rail length.

Your task is to decide how to build the fence given a list of point to point spacings, the price of each post and the lengths and prices of the specified rails. You may assume that the fence is built in a straight line.

Input will consist of a sequence of fences. For each fence the first line will contain two integers specifying the number of inter-point distances (*NP*) in the range 1 to 1000 inclusive and the cost of a post in dollars (*PP*). This will be followed by five (5) lines each containing a pair of integers representing the length of a rail in metres and its cost in dollars, both in the range 1 to 500 inclusive. These will be followed in turn by *NP* lines each containing a single integer representing an inter-point distance in metres. The end of input will be represented by a line containing two zeroes (0 0) for the values of *NP* and *PP*.

Output will consist of a single line for each fence. Each line will start with the words 'Fence number' followed by a space, the number of the fence (a running number starting at 1), a colon (':') and another space. This will be followed by the lowest price at which it is possible to build the fence followed by five integers specifying the numbers of the five given rail lengths necessary. All these numbers are to be separated by exactly one space.

Sample Input

```
1 4
1 4
2 4
3 4
4 3
5 10
3
2
1
3
1 5
1 1
2 2
3 3
4 4
5 5
3
1
1
1
3
0 0
```

Sample Output

```
Fence number 1: 15 0 1 1 1 0
Fence number 2: 12 0 0 0 1 1
```