New Zealand Programming Contest 2017

PROBLEM SET

PREAMBLE

Please note the following very important details relating to input and output:

- Read all input from the keyboard, i.e. use `stdin`, `System.in`, `cin` or equivalent. Input will be redirected from a file to form the input to your submission.

- Do NOT prompt for input as this will appear in your output and cause a submission to be judged as wrong.

- Write all output to the screen, i.e. use `stdout`, `System.out`, `cout` or equivalent. Do not write to `stderr`. Do NOT use, or even include, any module that allows direct manipulation of the screen, such as `conio`, `Crt` or anything similar.

- Output from your program is redirected to a file for later checking. Use of direct I/O means that such output is not redirected and hence cannot be checked. This could mean that a correct program is rejected! You have been warned.

- Unless otherwise stated, all *integers* will fit into a standard 32-bit computer word. If more than one integer appears on a line, they will be separated by spaces.

- An *uppercase letter* is a character in the sequence 'A' to 'Z'. A *lower case letter* is a character in the sequence 'a' to 'z'. A *letter* is either a lower case letter or an upper case letter.

- Unless otherwise stated, a *word* or a *name* is a continuous sequence of letters.

- Unless otherwise stated, a *string* is a continuous sequence of visible characters.

- Unless otherwise stated, words and names will contain no more than 60 characters, and strings will contain no more than 250 characters.

- If it is stated that 'a line contains no more than *n* characters', this does not include the character(s) specifying the end of line.

- Input files are often terminated by a 'sentinel' line, followed by an end of file marker. This line should not be processed.

PROBLEM A **ANGLES** **3 POINTS**

Angela is teaching her primary school pupils how to use a protractor (which hopefully you know is a device used to measure angles!). She has given each pupil a set of triangles and asked them to measure and record each of the three internal angles.

Angela would like you to write a simple program to check her pupil s results. Knowing that the internal angles of a triangle add up to 180º, which results should Angela check?

Input

You will be given a set of results from one class. It will start with a single integer, N, the number of pupils who have supplied readings. (0 < N <= 30). There will then follow N lines, each containing 3 positive integers separated by spaces. The numbers represent the measurements supplied by a pupil for the 3 angles of a triangle.

Output

For each set of readings, output the numbers as input followed by the word `Check` if they do not add up to 180, or `Seems OK` if they do.

Sample Input

```
10
59 60 60
42 68 70
65 65 50
120 31 28
61 61 58
78 61 41
85 85 9
64 82 35
29 102 49
120 30 30
```

Output for Sample Input

```
59 60 60 Check
42 68 70 Seems OK
65 65 50 Seems OK
120 31 28 Check
61 61 58 Seems OK
78 61 41 Seems OK
85 85 9 Check
64 82 35 Check
29 102 49 Seems OK
120 30 30 Seems OK
```

PROBLEM B **GOLF CROQUET** **3 POINTS**

In golf croquet doubles, two teams of 2 play each other. Each player has their own ball which they hit when it is their turn. A turn comprises one hit of the ball.

There are 6 hoops which are each played twice, once in each direction, in a prescribed order. A point is scored by the first team to make the current hoop. To make a hoop, one of the team s balls must be knocked through the current hoop in the right direction. The first team to score 7 points wins. If the game reaches 6-6, a final deciding hoop is played. Note that a team cannot score more than 7 points the game ends as soon as the 7th hoop is won.

The balls are played in a fixed order: blue, red, black (blue s partner) then yellow (red s partner). We can record the progress of a match by recording the outcome of each shot. The code we are using is:

S a standard shot that does not make a hoop

H a shot that makes a hoop for the team who played the shot

D a shot that makes 2 hoops for the team who played the shot an extremely unlikely scenario but it can happen! A team on 6 points will only score the first of the two hoops, of course.

O a shot that makes a hoop for an opponent s ball!

The last one is almost certainly an accident, but it happens far more times than players like!

Input

The input represents part or all of a single game of golf croquet. The first two lines each contain the name of a team, each consisting of one or more words. The name will be no more than 30 characters long. The first named team play blue and black. The third line is a single integer, `S`, which tells how many strokes are recorded, for a game that has started but which may not yet be completed. (0 < `S` <= 255)

The fourth line contains S upper case letter characters, each being one of the 4 characters defined above (S, H, D or O). Blue always plays the first shot followed in turn by red, black and yellow.

Output

Output a single line of text with the current score in the form

`team1name x team2name y.`

Follow this by a space then one of:

teamname `has won.`

teamname `is winning.`

`All square.`

Sample Input

```
Team Sally
The dragons
26
SSSSHSSSSSHSSSSSHSSSSSSHSS
```

Output for Sample Input

```
Team Sally 3 The dragons 1. Team Sally is winning.
```

Explanation

Team Sally are blue and black

The dragons are red and yellow

Shot 5 – Blue makes hoop 1 (1-0)

Shot 11 – Black makes hoop 2 (2-0)

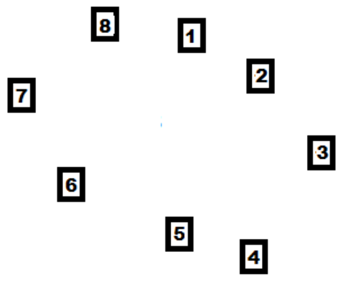
Shot 17 – Blue makes hoop 3 (3-0)

Shot 24 – Yellow makes hoop 4 (3-1)

New Zealand Programming Contest 2017

PROBLEM C **FITNESS** **3 POINTS**

Hilary is a fitness enthusiast. Near her home is a reserve which has 8 fitness stations set out in a roughly circular pattern, as shown. Every day, Hilary visits at least 5 of these stations as part of her daily exercise routine.



Hilary s daughter Catherine has been given the task of planning her mother s visits to the fitness stations. Catherine has produced a lot of daily plans, but would like them checked to see that they meet Hilary s requirements. This is your task! Do the plans visit at least 5 stations without repeats?

Input

Input consists of data for a single plan. The plan starts with a single one-digit number, S, which is the starting station (0 < S < 9). This is followed by a series of steps, each step being a letter followed by a positive one-digit number less than 8. The step tells Hilary which station to visit next. The letter for the step will be A (for anticlockwise) or C (for clockwise). The number will be how many stations to move. The last step will be #, which indicates there is no more data. Do not process this step.

Output

List all stations visited by the plan in the order they are visited, with a space between each station. If the plan does not visit at least 5 different stations, or if it visits a station more than once, the list should be followed by the word "`reject`".

Sample Input 1
```
4
A3
C5
A6
C5
#
```

Sample Input 2
```
5
C3
C3
A1
C3
C1
#
```

Output for Sample Input 1
```
4 1 6 8 5
```

Output for Sample Input 2
```
5 8 3 2 5 6 reject
```

Explanation
Rejected as it visits station 5 twice.

PROBLEM D **BEAUTIFUL MUSIC** **3 POINTS**

Recently discovered on a distant planet, a race of aliens has music a bit like ours (notes A, B, C, D, E, F and G) but without any sharps or flats. Like our music, after G they start again at A.

To these aliens, beautiful music is any sequence of notes where the difference between 1 note and the next is 2, 4 or 6 notes above the previous note. Anything else is discordant and causes them discomfort.

Input

Each line of input contains a set of N notes with no spaces between them (0 < N <= 21). Each note will be one of the upper case letters between A and G inclusive. The final line of input will be a # symbol on its own. Do not process this line.

You must assume that a note is the next note of that letter above the previous note.

Output

For each line of input, if the notes follow the rule for being beautiful music, your output should be
`That music is beautiful.`

Otherwise, the output should be

`Ouch! That hurts my ears.`

Sample Input	Output for Sample Input	Explanation
A	That music is beautiful.	Only 1 note must be OK.
C	That music is beautiful.	Only 1 note must be OK.
ACE	That music is beautiful.	Gaps are 2 and 2.
ABC	Ouch! That hurts my ears.	Gaps are only 1 and 1.
ACD	Ouch! That hurts my ears.	Gaps are 2 then 1.
CEB	That music is beautiful.	Gaps are 2 then 4.
EAF	Ouch! That hurts my ears.	Gaps are 3 then 5.
EBFA	That music is beautiful.	Gaps are 4 then 4 then 2.
#		Stop processing.

PROBLEM E **BYTE ME!** **10 POINTS**

Parity is an important concept in data transmission. Because the process is not error proof, parity is used to provide a check on whether or not data has been corrupted in transmission.

If even parity is being used, each byte will have an even number of 1 characters. If odd parity is being used, each byte will have an odd number of 1 characters.

In this problem, we are looking for a single bit that has been corrupted. To help you find it, the last byte is not part of the data being transmitted, but is a parity byte. Each bit of the parity byte will be used to make the corresponding bits in the data bytes odd or even depending on the parity being used.

Input

The first line of input is a single integer, N (3 <= N <= 10), the number of bytes of data to follow. The next N lines each contain a single byte of data consisting of 8 characters separated by spaces. Each character will be either 1 or 0.

There will be one further line of 8 characters (again 1 or 0) which will be the parity byte. In the parity byte, each bit is a parity bit for the corresponding bits in the preceding N lines, using the same parity as is used by the data bytes. The parity byte itself may not show the same parity as the data bytes.

Output

Output 3 lines of information about the data in the input.

Line 1: Either the word `Even` or the word `Odd` to describe the parity being used by the bytes which are not broken.

Line 2: `Byte <number> is broken`

Line 3: `Bit <number> is broken`

`<number>` is the number of the appropriate byte or bit, where the first of each is number 1.

Turn over for sample input and output

Sample Input 1

```
3
1 0 1 0 1 1 1 0
1 1 0 1 1 1 0 0
1 0 1 1 1 0 0 0
0 0 1 1 1 1 0 1
```

Output for Sample Input 1

```
Odd
Byte 3 is broken
Bit 5 is broken
```

Explanation 1
Bytes 1 and 2 have an odd number of 1s but byte 3 has an even number. So odd parity is being used but byte 3 is broken.
The parity byte gives all columns of bits an odd number of 1s except for 5 where they are even, so bit 5 is broken. Bit 5 of byte 3 is corrupt.

Sample Input 2

```
4
1 0 1 1 1 0 0 0
1 0 1 1 0 0 0 0
0 1 0 0 1 0 0 0
1 0 1 1 1 0 1 1
1 0 1 1 1 0 1 1
```

Output for Sample Input 2

```
Even
Byte 2 is broken
Bit 2 is broken
```

Explanation 2
The first 4 bytes all have an even number of 1s except for byte 2 which is the broken byte.
The parity byte gives all columns of bits an even number of 1s except for 2 where they are odd, so bit 2 is broken. Bit 2 of byte 2 is corrupt.

PROBLEM F **MUSICAL CHAIRS** **10 POINTS**

Musical chairs is a game frequently played at children s parties. Players are seated in a circle facing outwards. When the music starts, the players have to stand up and move clockwise round the chairs. One chair is removed, and when the music stops the players all have to try to sit down on one of the chairs. The player who does not manage to sit down is out, and the game continues until there is just one player left, who is the winner.

Input

The first line contains a single integer, N which is the number of players (1 < N <= 15). The next N lines have the names of the players.

The next line contains R, an integer which tells how many rounds are to be processed (0 < R < N). The next R lines each contain a pair of integers S and M, separated by a space. S is the number of the seat to be removed. Seats are numbered from 1 to the number of seats remaining in a clockwise direction.

M is the number of moves made before the music stops (0 < M <= 30). A move takes a player from one seat to the next in a clockwise direction. A move from the highest seat number takes a player to seat 1.

Output

After each round has taken place, output a line

```
<name> has been eliminated.
```

where <name> is the name of the person who does not find a seat. This will be the person who would have ended up at the seat which was removed.

At the end of the specified number of moves, output a line which either says

```
<name> has won.
```

where a single player remains, or

```
Players left are <name list>.
```

where the game is not yet finished.

`<name list>` contains the name of each player not yet eliminated in the same order as in the input. The names are separated by spaces.

Turn over for sample input and output

Sample Input 1

```
5
Anne
Bill
Chen
Di
Everet
4
3 6
2 8
1 5
2 6
```

Output for Sample Input 1

```
Bill has been eliminated.
Anne has been eliminated.
Chen has been eliminated.
Everet has been eliminated.
Di has won.
```

Sample Input 2

```
6
Ferdinand
Garcia
Henrietta
Isabel
Jacob
Kirstin
3
6 10
5 1
2 30
```

Output for Sample Input 2

```
Garcia has been eliminated.
Kirstin has been eliminated.
Jacob has been eliminated.
Players left are Ferdinand Henrietta Isabel.
```

PROBLEM G **LETTER COUNT** **10 POINTS**

Shona is a linguistics student who has an assignment to analyse letter distribution in sentences taken from various sources. Your task is to write a program to help her.

The problem is to plot a graph showing how many of each letter appear in the sentence supplied. Upper and lower case letters are considered the same letter. Characters that are not letters of the English alphabet are ignored.

Input

The input consists of a single line of text of no more than 250 characters. At least one character will be a letter.

Output

Output consists of a graph with 26 horizontal bars. The letters of the alphabet (upper case) form the y axis, in alphabetical order from top to bottom. Each is followed by a space, a vertical bar, another space then a star for every time that letter appears in the supplied text. There are no spaces between the stars.

Sample Input

```
This is a piece of sample text for the New Zealand Programming Contest 2017.
Plot a graph to show its letter frequency.
```

NOTE This is a single line of text which wraps round on the page, but has no line break in it.

Output for Sample Input

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

PROBLEM H **LEAGUE TABLES** **10 POINTS**

League football (known in some circles as soccer) has been played in England since 1888 and is the most popular winter game through most of Europe, just as rugby is in New Zealand. English newspapers and many Websites publish the latest results and the current tables.

With English football, each team plays every other team both home and away, and at the end of the season, the team with most points wins the title. Points are awarded for winning (3 points) or drawing (1 point each). Teams level on points are separated by the larger goal difference, that is goals for (ie scored) minus goals against (ie conceded). If this is level, the team who has scored more goals is placed first.

In this problem you will be given a list of teams and their current record, followed by a list of matches. You have to update the record of each team who has a result recorded and output a correctly sorted league table.

Input

The first line contains a single integer, T (8 <= T <= 30), which is the number of teams in the league. The next 2 x T lines each contain the current record of one team as follows:

```
<name>
<games played> <wins> <draws> <losses> <goals for> <goals against> <points>
```

The first line contains the team name which consists of one or more words, which may contain numbers. No name will be longer than 30 characters.

All numbers on the second line are non-negative integers.

The next line contains a single integer, G (0 < G <= (T/2)), which is the number of games recorded. There then follow 4 x G lines, each containing data on one game as follows:

```
<home team name>
<home team score>
<away team name>
<away team score>
```

The two teams will both be from the preceding list of teams. The scores will each be a non-negative integer less than 20.

Output

T lines giving the updated record of each team from the input, name followed by data all on 1 line. Teams are to be sorted by highest points, then highest goal difference, then most goals scored, then alphabetical order (case insensitive, numbers before letters).

Turn over for sample input and output

Sample Input

Input is shown in 2 columns to keep it on 1 page.

```
18
Aston Villa
33 21 6 6 74 34 69
Sheffield United
34 18 12 4 63 33 66
Sunderland
33 18 3 12 47 34 57
Wolverhampton Wanderers
33 15 9 9 47 35 54
Newcastle United
34 13 10 11 53 43 49
Manchester City
33 13 8 12 49 41 47
Nottingham Forest
34 13 8 13 56 55 47
Derby County
33 13 8 12 43 42 47
Stoke City
34 13 8 13 37 45 47
Everton
33 13 7 13 46 30 46
Bury
33 13 6 14 40 57 45
Liverpool
33 13 5 15 47 45 44
Blackburn Rovers
33 12 4 17 47 60 40
West Bromwich Albion
34 11 8 15 43 51 41
Preston North End
33 12 4 17 37 45 40
Notts County
34 9 11 14 46 60 38
Burnley
34 11 5 18 34 54 38
Glossop
34 4 10 20 31 74 22
5
Aston Villa
3
Preston North End
1
Blackburn Rovers
2
Wolverhampton Wanderers
1
Derby County
2
Everton
1
Liverpool
2
Bury
0
Sunderland
3
Manchester City
1
```

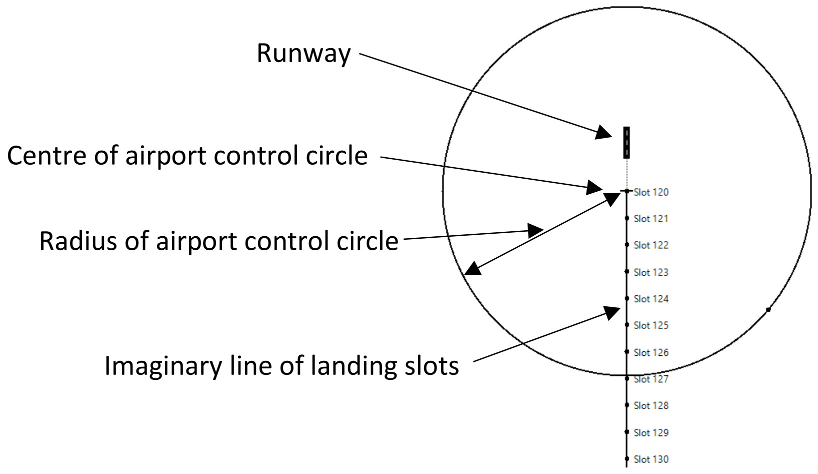
Output for Sample Input is on the next page.

Output for Sample Input

```
Aston Villa 34 22 6 6 77 35 72
Sheffield United 34 18 12 4 63 33 66
Sunderland 34 19 3 12 50 35 60
Wolverhampton Wanderers 34 15 9 10 48 37 54
Derby County 34 14 8 12 45 43 50
Newcastle United 34 13 10 11 53 43 49
Manchester City 34 13 8 13 50 44 47
Liverpool 34 14 5 15 49 45 47
Nottingham Forest 34 13 8 13 56 55 47
Stoke City 34 13 8 13 37 45 47
Everton 34 13 7 14 47 32 46
Bury 34 13 6 15 40 59 45
Blackburn Rovers 34 13 4 17 49 61 43
West Bromwich Albion 34 11 8 15 43 51 41
Preston North End 34 12 4 18 38 48 40
Notts County 34 9 11 14 46 60 38
Burnley 34 11 5 18 34 54 38
Glossop 34 4 10 20 31 74 22
```

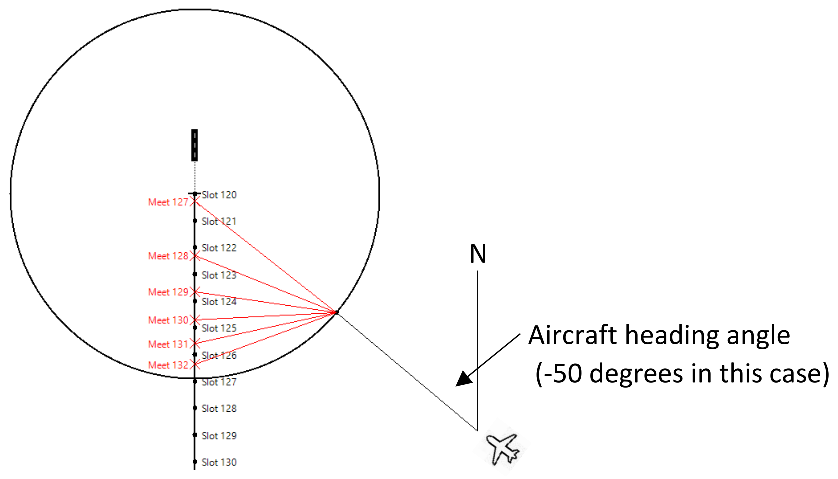
PROBLEM I **PLANES** **30 POINTS**

The Airport at NZPC headquarters is small by international standards, but the number of flights is steadily increasing and has reached the point at which computer help with air traffic control is needed. You have been asked to write part of the control algorithm. The airport is responsible for aircraft in its immediate vicinity, and in particular for managing landing paths. Here is a diagram of the airport environment.


Runway
Centre of airport control circle
Radius of airport control circle
Imaginary line of landing slots
Slot 120
Slot 121
Slot 122
Slot 123
Slot 124
Slot 125
Slot 126
Slot 127
Slot 128
Slot 129
Slot 130

The environment is a large circular area centred about 5 km south of the runway. Incoming aircraft fly directly toward the centre from distant places (incoming from easterly and westerly directions – never flying due north or south). When they reach the edge of the control circle they must be given a 'landing slot' on the landing path. Incoming aircraft will all be travelling at a fixed speed of 400 km/hour. Imagine the landing path as a huge invisible 'conveyor belt' in the sky, extending south from the control area centre. At midnight each day the conveyor belt starts with slot 0 at the centre, and slots 1, 2, … each spaced 30 seconds of flying time apart to the south. The 'conveyor' moves north at aircraft flying speed. The diagram above shows the conveyor belt at 1am in the morning – with slot 120 just crossing the centre. When an aircraft is allocated its slot, it turns to fly so as to exactly meet its 'slot' on the landing path, and then turns to align with the landing path and proceeds to touch down shortly after crossing the centre. Usually there are several possible slots available for an aircraft, the allocation rules just requiring that the aircraft must join its slot south of the centre and inside the control circle. The diagram on the next page shows some possibilities.

Here an aircraft approaches with heading -50 degrees (+ve angle is clockwise from North). It arrives at the edge of the (in this case 23 km radius) control circle at exactly 1am (just as slot 120 passes the centre). The first slot it could meet is 127, and it would meet that slot at the point shown approximately 3.5 minutes after arriving at the edge. Turning left at successively greater angles would allow it to meet slots 128, 129, … through to 132 at the points shown. If it turned to meet slot 133, the meeting point would be outside the control circle and that would therefore not be allowed.



The algorithm for selecting landing slots processes the aircraft in the order in which their data is received (which will be a non-decreasing time order). Each aircraft is allocated to the first slot it can reach which has not yet been used for another aircraft. Note that the algorithm does not check to see if flight paths intersect – another part of the system will do safety checks.

Input

Input will consist of a series of landing scenarios. The first line of the input holds a single integer P being the number of scenarios. Then for each scenario there will be a line holding two values: R ($10 <= R <= 100$) being the radius of the air traffic control circle in km for the scenario, and N ($1 <= N <= 30$) being the number of aircraft landing in this scenario; followed by N lines, each with two values t_i ($0 <= t_i <= 20$) being the arrival time of the aircraft at the edge of the control circle in hours after midnight and b_i (either $-90 <= b_i <= -10$ or $10 <= b_i <= 90$) being the heading angle in degrees of the aircraft on arrival. All values will be integer except for the times t_i which will be given with four decimal places. All lines are formatted with single spaces between numbers and no leading or trailing spaces. For a given scenario the times will be non-decreasing.

Output

For each scenario your program should output one line with the text "Scenario" and the number of the problem (counting from 1), followed by N lines, one for each aircraft (in the same order as given in the input). For each aircraft you should output the slot number allocated to the plane, the distance from the centre (in km) at which the plane meets the landing path, and the time at which the aircraft will reach the centre of the control area (to commence final descent and landing).

Follow the format shown in the sample output. The distance and time values should be displayed rounded with four decimals. Sample and test data has been chosen to avoid marginal rounding values. You may assume that at least one landing slot will be available for each aircraft.

Sample Input

```
1
23 3
1.0000 -50
1.0000 90
1.0005 -80
```

Sample Output

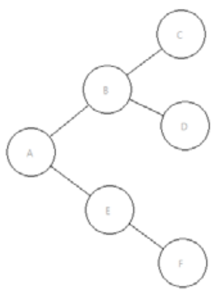
```
Scenario 1
Slot 127, joining at 0.9033km south, final approach at 1.0583
Slot 128, joining at 3.4146km south, final approach at 1.0667
Slot 129, joining at 6.9565km south, final approach at 1.0750
```

PROBLEM J **TREES** **30 POINTS**

Modern art is something that everyone has an opinion about – you love it or you hate it. Rarely discussed however is what aficionados consider to be the pinnacle of human (and computer) artistic achievement – ASCII art. In this problem you are (or rather your computer is) asked to rise to the lofty aesthetic levels of this amazing art form and draw a binary tree.

Consider the tree shown below on the left. It has six nodes, each labelled with an upper case letter of the alphabet. Node A is the root of the tree. It has two sub-trees, the first shown above to the right of A and the second below to the right. The tree is binary in the sense that each node may have at most two sub-trees, however it is not balanced – E has no first sub-tree, only a second.

The same tree is shown on the right in ASCII art. Nodes are shown using their letters. Single forward and backward slash characters are used for edges. Where a subtree doesn't fit into the space above or below its parent node the connecting line is extended with vertical bar characters directly above or below the node character, as required. See sample output for examples.



```
    C
   /
  B
 / \
|   D
A
 \
  E
   \
    F
```

Input

Input will consist of a series of trees. The first line of the input holds a single integer T, being the number of trees (1 <= T <= 100). Then for each tree, there is a line of input describing the tree in prefix format – for a node the letter of the node is output first, followed by the coded first subtree, and then the coded second subtree. Each node is output with two subtrees. If a subtree is empty it is output as a '@' character. The input for the tree drawn above is ABC@@D@@E@F@@
Each input tree description will be no longer than 100 characters. Note that node letters may be repeated and may not be in alphabetic order.

Output

For each tree your program should output one line with the text "Graph" and the number of the tree (counting from 1). This should be followed by the ASCII tree drawn as described. Lines should not have trailing blank characters. Output one blank line between trees. Note that the sample output is presented twice on the next page – on the left as it should appear, and on the right with space characters replaced by periods. This is provided to help clarify the format.

Sample Input

```
2
ABC@@D@@E@F@@
AGH@I@J@K@L@@@MN@@@
```

Sample Output

```
Graph 1
    C
   /
  B
 / \
|   D
A
 \
  E
   \
    F

Graph 2
    H
   / \
  |   I
  |    \
  |     J
  |      \
  |       K
  |        \
  |         L
  G
 /
A
|   N
 \ /
  M
```

Version of output showing spaces as periods

```
Graph 1
....C
.../
..B
./.\
|...D
A
.\
..E
...\
....F

Graph 2
....H
.../.\
..|...I
..|....\
..|.....J
..|......\
..|.......K
..|........\
..|.........L
..G
./
A
|...N
.\./
..M
```

PROBLEM K **PACKING** **30 POINTS**

It was bound to happen. Modernisation has reached the North Pole. Faced with escalating costs for feeding Santa Claus and the reindeer, and serious difficulties with security, NP Management has decided to do away with the traditional sleigh and adopt delivery by drone (magic, superfast drone). Lack of investment capital means that the new system will start small, and hopefully grow in the years to come. For the first test run in 2017 there will be only two drones and they will have limited carrying capacity. PR is, of course, all important. There will be disappointment, and NP Management has decided to focus on delivering only the most expensive toys to the richest children, so as to focus the worst of the disappointment on those who have the greatest experience of coping (the poor).

Choosing the presents to deliver is your problem. You are being asked to develop an algorithm to select the cargo to deliver, given weight limits for each of the drones and a list of candidate presents with weights and values. Your goal is to maximise the value of gifts delivered.

Input

Input will consist of a series of problems. The first line of the input holds a single integer P being the number of problems. Then for each problem there will be three lines of input. The first line holds three integers: N (1 <= N <= 100) being the number of candidate presents; W1 and W2 (1 <= W1, W2 <= 1000) being the weight limits of the two drones respectively. The second line holds N integers (1 <= w_i <= 100) being the weights of each of the candidate presents and the third line holds N integers (1 <= v_i <= 100) being the values of the presents (in thousand dollar units). All lines are formatted with single spaces between numbers and no leading or trailing spaces.

Output

For each problem your program should output one line with the text “Problem “ and the number of the problem (counting from 1) followed by a colon, a space and the total value of presents shipped by the drone pair.

Sample Input

```
2
4 9 4
3 4 5 6
5 7 9 10
6 9 11
3 4 5 6 3 4
2 3 4 5 3 3
```

Sample Output

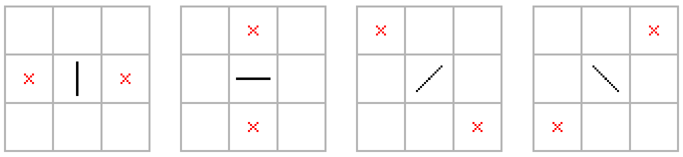
```
Problem 1: 22
Problem 2: 16
```

PROBLEM L DOMINO KILLING 30 POINTS

One of my favourite childhood games is dominoes – not the original game where tiles are placed flat on a table with numbers matching up (which is also fun), but the game where the tiles are placed upright in a chain, then the first tile is pushed and all of them topple.

Sometimes the dominoes are not positioned well and not all of them will topple. Given the layout of the dominoes, can you work out how many will topple when a particular domino is pushed?

The dominoes are placed on a board at integer coordinates. They have four possible orientations, as shown in the following diagrams (viewed from above):



The black lines are the tops of the dominoes, and the crosses show where they would fall if pushed. The dimensions of the dominoes are such that when they topple they will only push the domino at the marked location (if there is one there).

If a domino hits a second domino that is at an angle of 90 degrees to the first, the second domino will not be toppled (but the first is still counted as toppled).

The positions on the board will be described using x and y coordinates, with the x-axis going from left to right and the y-axis going from top to bottom in the diagrams.

The name of this problem comes from the translation by a popular translation tool of “ドミノ倒し”, the name of this game in Japanese.

Input

Input will consist of a sequence of problems. The first line for each problem will have four space-separated integers: N, the number of dominoes ($1 \le N \le 100{,}000$); S, the 0-based index of the first domino pushed ($0 \le S < N$); then Fx and Fy, the direction of the push as a vector (Fx and Fy will be -1, 0, or 1). The following N lines will describe the dominoes. Each line will have two integers and a symbol separated by spaces. The two integers are the x and y coordinates of the domino ($0 \le x, y, \le 100{,}000$), and the symbol is one of `|`, `-`, `/`, or `\`, giving the orientation of the domino when viewed from above.

The vector `(Fx, Fy)` will be one of the two directions that domino S can be pushed; for example, if the orientation of domino S when viewed from above is `|`, then `(Fx, Fy)` will be either `(1, 0)` or `(-1, 0)`.

End of input will be denoted by four zeroes and should not be processed.

Output

For each problem, output a single integer with the maximum number of dominoes that can be toppled by pushing the first domino in the input.

Sample Input

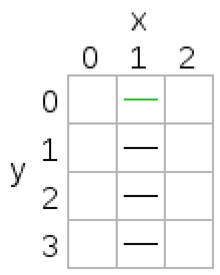
```
4 0 0 1
1 0 -
1 1 -
1 2 -
1 3 -
8 0 0 1
0 1 -
0 2 /
1 3 |
2 3 \
3 2 -
3 1 /
2 0 |
1 0 \
6 1 1 0
0 0 |
1 0 |
2 0 /
3 0 |
3 1 /
4 2 \
0 0 0 0
```

Output for Sample Input

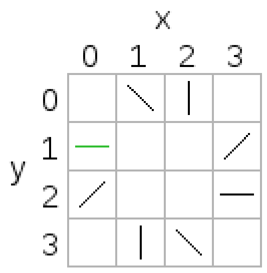
```
4
8
3
```

Explanation is on the following page

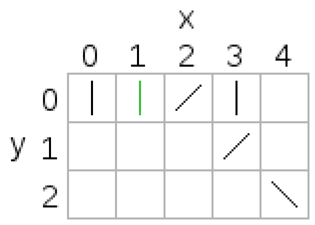
Explanation



Problem 1: The domino pushed is the top one. It is pushed in the direction `(0, 1)` which is down in the diagram, so it will topple the next domino which will topple the next one and so on. All four dominoes will be toppled.



Problem 2: The domino at `(0, 1)` is pushed in the direction `(0, 1)` which down in the diagram. It will topple the domino at `(0, 2)`, which will topple the domino at `(1, 3)`, and so on until all the dominoes topple. When the domino at `(1, 0)` topples, nothing further will happen because the domino at `(0, 1)` has already toppled.



Problem 3: When the domino at `(1, 0)` is pushed to the right, it will only topple the dominoes at `(2, 0)` and `(3, 1)`. The domino at `(4, 2)` will be pushed by the domino at `(3, 1)` but won't topple because it is at a 90 degree angle.

PROBLEM M **INFINITE TREES** **100 POINTS**

Type checking is the process of verifying that a computer program satisfies the typing rules of a programming language. In some languages this is very difficult, and just checking if two types are equal can be tricky!

For this problem, a type will be represented as a tree made up of nodes. Each node has zero or more child nodes, corresponding to component types. Here is an example of a tree:

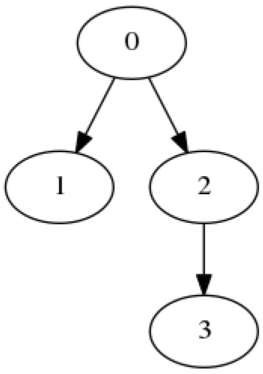


Figure 1: A simple finite tree. The root is node 0 and its children are nodes 1 and 2. Node 1 does not have any children, and node 2 has one child (node 3).

Types are allowed to be recursive. This means that a node can have any node as a child, including its parent and even itself. The result is an infinite tree:

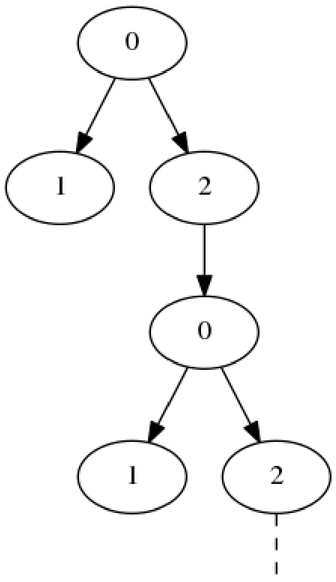


Figure 2: An infinite tree. Node 0 has node 2 as a child, which has node 0 as a child, which has node 2 as a child, and so on. The tree keeps going forever, and has been truncated to fit on this page.

Given two possibly infinite trees, your task is to determine whether or not they have the same structure. Two nodes A and B have the same structure if

- They have the same number of children, and
- The i'th child of node A has the same structure as the i'th child of node B, for each child index i.

Two trees have the same structure if their root nodes have the same structure.

Input

Input will consist of a sequence of problems. The first line for each problem will have two space-separated integers, N and M, the number of nodes in the two trees (1 ≤ N, M ≤ 100,000).

The following N lines will describe the first tree. The i'th of these lines (counting from 0) will consist of a number of space-separated integers describing node i. The first integer will be c[i], the number of children that node i has (0 ≤ c[i]; sum of c[i]'s ≤ 100,000). The rest of the line will contain c[i] space-separated integers, the children of node i. The following M lines will describe the second tree similarly. The root of each tree is always node 0.

Each problem will be followed by a blank line. End of input will be denoted by a line with two zeroes (which should not be processed).

Output

For each problem, output "YES" if the two trees have the same structure, or "NO" if they don't.

Sample input and output are on the following page

Sample Input

```
4 4
2 1 2
0
1 3
0
2 2 1
0
1 3
0

3 4
2 1 2
0
1 0
2 1 2
0
1 3
2 1 2

3 1
2 1 1
2 2 2
2 0 0
2 0 0

0 0
```

Output for Sample Input

```
NO
YES
YES
```

Explanation

For the first problem, Figure 1 shows the first tree and the other tree is its mirror image. They have different structures, because the order of the children is significant.

Figure 2 shows the first tree in the second problem. The other tree is represented using more nodes but has the same structure.

Time Limit: 10 seconds　　　　**Competition Day**　　　　100 Points

Problem Statement

Matt, Nick and Tim are all best friends who flat together and regularly compete in programming competitions. A big competition is coming up, and as Matt recently learned C# he would like to show off his skills and compete against Nick and Tim. However Nick and Tim are certain their combined Fortran 95 skill will triumph against Matt.

Nick suggests that on the competition day they take a different route to the competition than Matt does to get in his head. To ensure Matt sees this different route they take, they will leave at the same time from their flat. Unfortunately for Nick and Tim, Matt is one step ahead and has already planned out the fastest route to the competition from their flat!

Tim hates to waste petrol on the same scenery, and therefore they will not visit the same place twice on their route. The potential routes from the flat to the competition will include N junctions and M directed roads connecting these junctions. Each road has a time T associated with it (in minutes) for how long it takes to drive down and these measurements are well known. The flat and competition site are located at different junctions.

How long after Matt will Nick and Tim arrive (in minutes) if they take the fastest route which is different from Matt's route?

Input

Input will begin with a line containing four integers, N, M, F, C. $2 \leq N \leq 8000$, $1 \leq M \leq 8000$, $1 \leq F, C \leq N$
N is is the number of junctions and M is the number of roads. F and C represent the junction number of the flat and competition site respectively.
M lines will follow with three integers i, j and T, $1 \leq i, j \leq N$ representing a unidirectional road from junction i to junction j taking T minutes. T is a positive integer. $0 < T \leq 2 \times 10^5$
Input will contain many test cases.
The last case will be denoted by N and M being 0.
There will always be a path from the flat to the competition site.
No two roads connect the same pair of junctions in the same direction. No road connects a junction to itself.

Output

Output how many minutes Nick and Tim will arrive after Matt.
If Nick and Tim can not take a different route, print "Matt wins." as they have failed to get in his head.

Sample Data

Input

```
2 1 1 2
1 2 1
3 3 1 3
1 2 1
2 3 1
1 3 3
0 0 0 0
```

Output

```
Matt wins.
1
```

PROBLEM O **SCHOOL PAIRING** **100 POINTS**

The school of Auckbury has one of the highest admission rates in the world. In some years its annual enrolment almost reached one million! This school highly values the ability to succeed at programming competitions, and therefore on admission each student is graded on how well they can compete. The grade is an integer in the range -10^9 to 10^9.

Each year, the newly admitted students are assigned a number from 1 to N, where N is the number of students admitted that year. This is so the new students can easily be identified for the "Freshman competitions". These competitions are held very frequently throughout the year and only apply to the recently admitted. When a competition occurs, the staff computer will generate two numbers between 1 and N, representing student numbers.

The competition then proceeds by students between these two inclusively getting into teams of two, if possible, to compete against other teams. However, the principal prides himself for how fair he is and therefore will only allow a team of two to compete if the team has a joint skill level (by summing the two) equal to K, which is determined at the start of each day.

Given the M competitions held each year, the principal would like to know how many teams can be formed between the two numbers generated.

Input

Input will begin with a line containing three integers, N, M and K. $1 \le N,M,K \le 10^6$
Following this will be a line containing all the N students skill levels. $-10^9 \le Ni \le 10^9$
Followed by M pairs of student numbers i and j, $1 \le i,j \le N$ representing the two numbers generated.
Input will contain many test cases, representing each competition, the last case being denoted by N, M and K all being 0.

Output

Output the number of distinct teams of students that form a total skill level of K between each pair of student numbers in order of queries received. After each test case print a blank line.

Sample Input

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

Sample Output

```
1
0
```

Explanation

There are four students lined up on the field, and the principal wishes teams to have a combined skill level of 5. He first points at the first and fourth student, encompassing all four inclusively. There exists one team in this range, consisting of the first two students which has a combined skill of 5. He then points at the third and fourth students, between which there exists no team of two with a total skill of 5.