Problem A. Cyclic Troubles

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 megabytes

A package has been delivered to the King of Berland containing a game called "Bercycles". It's an entertaining single-player game with simple rules:

- You're given a rectangular field consisting of R rows and C columns, which result in $R * C$ cells total. Each cell contains a lowercase Latin letter and an arrow pointing one of the four following directions: left, right, up, or down.
- You choose the cell at which you wish to start the game and make a number of moves. At each move you have to write down the letter from the current cell and move one cell in the direction shown by the arrow at this cell. You can stop moving any time you want. If you leave the field, the process stops automatically and you're not allowed to move anymore.
- The letters you write down during your movements form a word that you *read* during the game.

After reading the rules the King of Berland realized that he is in trouble. He is given a number of words and for each word he is to determine whether it is possible to play the game and *read* this word. Please help the King to solve this challenging problem.

Input

The first line of input file contains integer numbers R and C ($1 \le R \le 30$, $1 \le C \le 30$) — number of rows and columns correspondingly. The following R lines describe arrows on the field. Each of these R lines contains exactly C characters. Each character denotes a direction:

- '<' character - LEFT
- '>' character - RIGHT
- '^' character - UP
- 'v' character - DOWN

Then R more lines follow, describing letters on the field. Each of these R lines contains exactly C characters. Each character is a lowercase Latin letter 'a'-'z'.

Then one more line follows containing a single integer number Q ($1 \le Q \le 50$) — number of queries. Each of the following Q lines contains a word for which you should solve the problem.

To reduce the size of input files, these Q words are given to you in a compressed form. A compressed form of a word may contain non-empty letter sequence F enclosed in parentheses, followed by a positive integer number K. It means that the fragment F should be repeated K times.

For example, the string "a(xy)2y(ab)3abz" is one of the compressed forms of the string "axyxyyababababz". Note: parentheses may not be enclosed into each other, so "a(xy)2y((ab)2)2z" is not a compressed form of the same word. Leading zeroes are not allowed as well in a compressed form, so no queries like "(ab)02" will be given to you.

Input file will not contain any malformed compressed words. The length of each compressed query will be between 1 and 2000 characters. When decompressed, the length of each query will be no longer than 10^9 characters.

Output

For each query you should print a single line to the output file. If the corresponding compressed word can be read on the field, then the line should contain "YES (X,Y)", where X and Y correspond to the row and column in the field where you should start in order to read the given word. Rows and columns are numbered starting from 1.

If the word can't be found in the field, you should print a single word "NO" in the line. If there is more than one solution, print the one with a minimal row X. If there is more than one solution with the same minimal row, print one that minimizes column Y. Please see sample output for further clarifications on the output file format.

Examples

stdin	stdout
2 4 >>>v <^<< abcd efgh 6 abcdhgf bcdhgf (bcdhgf)100 a(bcdhgf)100bc b(cdhgfbc)1d hello	YES (1,1) YES (1,2) YES (1,2) YES (1,1) YES (1,2) NO

Problem B. Bergamot Problem

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 megabytes

Berland alphabet consists of N letters, which are first N latin letters. Citizens of Berland use simple abbreviation scheme for SMS writing purposes. Each word is abbreviated to the sequence of two symbols — its first and last letters. For example, the word "`hello`" is abbreviated to "`ho`". Unfortunately, for single-letter words this rule leads to a word lengthening. For example, word "`a`" will be written as "`aa`".

Vocabulary used for SMS creation consists of only two-character words. The numerous studies of well-known scientists showed that the vocabulary consists of M distinct words.

Bergamot company is ready to put their new cell phone in production. The new model is planned to have the revolutionary system of a quick text input (somewhat similar to the famous T9). Keyboard of the phone will consist of K buttons. Each button will have one or several letters from placed on it. Each letter can be placed on exactly one button of the phone. In order to enter a word, a user should press two buttons on the phone one after another: button with the first letter of the word, and then button with the last letter of the word. Placement of letters on the keyboard is called *correct* for a given vocabulary, if there are no words from vocabulary that can be entered using the same two-buttons sequence.

For example, if the vocabulary consists of three words "`aa`", "`ab`" and "`bc`", two buttons are enough for creating the keyboard. The first button will have "`a`" and "`c`", the second button — "`b`". The placement like "`a`" and "`b`" on the first button, "`c`" on the second button will not work, because words "`aa`" and "`ab`" can be entered using the same sequence of buttons.

Bergamot engineers have got a problem finding optimal placement of Berland alphabet letters on the keyboard of the phone. The *optimal* placement is any correct placement with the minimal possible number of buttons used for the given Berland alphabet.

Input

The first line of the input contains two integer numbers N, M ($1 \le N \le 13$; $0 \le M \le 50$). The following M lines contain words from the vocabulary — one word per line. Each word consists of exactly two lowercase Latin letters. Only first N letters from Latin alphabet can be used. All words in the vocabulary are different.

Output

The first line of the output file should contain the minimal possible number of the keyboard buttons K for the optimal letters placement. The placement itself should be printed to the following K lines. The $i + 1^{th}$ line of the output file should contain letters placed on the i^{th} button. The order of letters within each line, and the order of lines describing the placement don't matter. If there are several solutions, you may output any of them.

Examples

stdin	stdout
3 3 ab aa bc	2 ac b
4 2 ab cd	2 cb ad
5 4 aa bb cc dd	4 ae b c d

Problem C. Berhatton

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	64 megabytes

Berhatton is one of the most picturesque districts of Berland. All streets in Berhatton are parallel to the coordinate axes. All streets are really long, so you can consider them infinite. Any point with integer coordinates is an intersection of two streets. So the whole plane is divided by streets into one-unit squares. All squares are occupied by the sky-scrappers. Due to this fact, the distance between point (x_1, y_1) and point (x_2, y_2) is equal to $|x_1 - x_2| + |y_1 - y_2|$. BerPizza company has been working in Berhatton for a while and has N pizza huts. In each pizza hut they can make any kind of pizza from the menu and deliver it to the customer. Unfortunately, the company doesn't have enough funds to buy cars or even bikes for pizza delivery, so pizza boys have to deliver pizza on foot. Pizza boys from different huts run with the different speed. Everybody knows the famous motto of BerPizza "Deliver pizza in 10 minutes", so pizza boys have to move fast.

Nowadays BerPizza company has a hard time, because hamburgers become more and more popular. BerPizza has even decided to close some of their huts because of financial problems. BerPizza president has made a rush decision that the pizza hut A can be closed if pizza boys from at least K other pizza huts can reach it in 10 minutes. Your task is to find numbers of pizza huts, which can be closed.

Input

The first line of the input contains two integer numbers N and K ($2 \le N \le 10^5$, $1 \le K \le N - 1$). The following N lines contain coordinates of pizza huts x, y, and number w — the amount of unit segments pizza boy can cover in 10 minutes ($0 \le x \le 10^9$, $0 \le y \le 10^9$, $1 \le w \le 10^9$).

Output

Print the number of pizza huts T which can be closed to the first line of the output file. The next line should contain T numbers — the numbers of pizza huts that need to be closed. Pizza hut numbers should be printed in increasing order. Pizza huts are numbered from 1 to N in the order they are given in the input file.

Examples

stdin	stdout
2 1 0 0 1 1 0 1	2 1 2
3 2 0 0 1 1 0 1 10 3 1	0
3 2 0 0 100 13 17 5 10 10 10	1 2

Problem D. "Binary Cat" Club

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	64 megabytes

"Binary Cat" club is extremely popular among Berland citizens. Club is a pretty crowded place and some unpleasant incidents happen there from time to time. So the club management monitors their visitors thoroughly. Usual measures like "face control" are accompanied with so-called "logging". Each time visitor enters the club, the "`+ name`" record is added to the log. `name` is the name of the person who just entered. Similarly the "`- name`" record is added when a person leaves the club. From time to time security adds records like "`= visitors`", where `visitors` is the number of the club guests present at the moment. At the moment the club was opened, it was empty. All records are added in the chronological order. If a person entered and exited the club several times, all these events are recorded. At the moment the club is closed some guests may still be there to participate in after-party. The club is big enough to contain any number of visitors at the same time.

However all this security precautions didn't help. Recently a big scandal happened in the club. The scandal was about the discussion of implementation details of Malhotra-Kumar-Maheshwari algorithm. Police requested the log of the events for the night of the incident from the club management. During logs archive inspection club personnel found out that logs for the night of the incident are not complete due to technical reasons. Some of the records in the log are missing, but all the remaining records are preserved in their original order.

Now club management wants to "polish" the log a bit, so that it looks plausible. They believe that the log looks plausible, if for the sequence of records in the log there is a sequence of actions which can happen in real life. For example, if the log contains a record that somebody leaves the club, the record about the same person entering the club should be present earlier in the log. Or, a person cannot enter the club the second time, if she did not leave the club after the first entry. The records "`= visitors`" should represent the actual number of guests in the club.

The club managers would like to add some random records to the log in such a way, that the relative order of existing records is preserved and the final log is plausible. Your task is to find minimal possible number of records to be added to the log and to create one of the variants of "polished" log.

Input

The first line of the input contains integer number N ($1 \le N \le 200$), where N is the number of existing records in the log. Following N lines contain records of the log, one per line. Each record has the form "`+ name`", "`- name`" and "`= visitors`". The value of `visitors` is between 0 and 100, inclusive. All names have length between 1 and 16 characters and consist of only lowercase Latin letters.

Output

Write in the first line M, the minimal possible number of records in "polished" log. The following M lines should contain the records of the "polished" log in the same format they were given in the input. The limitations on the format of names and the number of visitors remain. If there are several solutions, choose any of them.

Examples

stdin	stdout
3 + andrew + ilya - mike	4 + andrew + ilya + mike - mike
1 = 2	3 + first + second = 2
3 - silentbob - silentbob = 1	6 + silentbob - silentbob + silentbob - silentbob + silentbob = 1

Problem E. Dance it up!

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	64 megabytes

Dance Dance Revolution is a game machine which forces a player to dance. The goal of the game is to step the dance-floor buttons in appropriate time. There are four buttons on the floor: LEFT, UP, RIGHT and DOWN.

Petya is a big DDR fun and spends all his spare time dancing on it. But he has a problem: he doesn't have enough of stamina. Therefore he asks you to write a program which can compute the optimal sequence of actions for a song in order to minimize the energy required to dance it.

The song for the DDR machine is divided into several equal time intervals (beats). For each beat it shows the buttons to be pressed. It is allowed to press non-required buttons (there is no penalty for doing this), but all required buttons should be pressed. For now, Petya is not a very good dancer and you can assume that each beat requires at most one button pressed.

The initial player position is "left leg on the LEFT button and right leg on the RIGHT button". Each beat a player is allowed to follow one of these patterns:

- Not to change legs position. Not to press buttons. This action costs no energy.
- Not to change legs position. Press the button on which one of the legs is. This action costs 1 point of energy.
- Move the leg, which didn't press a button at the previous beat, to a vacant button and press it. Another leg doesn't change position. This action costs 3 points of energy.
- Move the leg, which pressed a button at the previous beat, to a vacant button and press it. Another leg doesn't change position. This action costs 9 points of energy.
- Jump with both legs to any two different buttons and press both of them. This action costs 10 points of energy.

After testing a Prove-Of-Concept version of the program, Vasya realized that if program instructs him to move into position "left leg on RIGHT button and right leg on LEFT button" he can't see monitor with further instructions and fails immediately. So, this position is illegal.

Input

The first line of the input file contains integer number N — length of the song in beats ($1 \le N \le 1000$). The second line consists of exactly N characters and contains instructions for each beat. Characters 'L', 'U', 'R' or 'D' mean that the required button for a beat is LEFT, UP, RIGHT or DOWN correspondingly. Character 'N' means that there are no required buttons at this beat.

Output

In the first line output the amount of energy required for stepping the song. After that, output exactly N lines with two characters in each. Each line should contain the optimal legs position (the first character for the left leg, the second character — for the second) for the corresponding beat. Use 'L', 'U', 'R' and 'D' characters for LEFT, UP, RIGHT and DOWN buttons correspondingly. If there are several solutions you can output any of them.

Examples

stdin	stdout
6 RULURL	14 LR UR UL UL UR LR
9 LURLDRLNL	25 LR LU RU DL DL DR LR LR LR

Problem F. Text Editor

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	64 megabytes

The simplest text editor "Open Word" allows to create and edit only one word. The editor processes keys 'a' – 'z', and also 'L' (to the left) and 'R' (to the right). After starting his work the editor immediately creates an empty word and sets its cursor to the left-most position. When one of keys 'a' – 'z' is pressed, the text editor inserts corresponding symbol just after the cursor. After that a cursor moves one position to the right in such a way that it is placed just after new symbol. When key 'L' or 'R' is pressed, the cursor moves one position to the left or to the right respectively. If the cursor can't be moved because it is placed at the left-most or right-most position the command is ignored.

Developers of "Open Word" didn't think about the effectiveness so the editor is working slowly if a lot of keys have been pressed.

Your task is to write a program that can process a sequence of key pressings emulating this editor and output result string.

Input

The input file contains one string which consists of symbols 'a' – 'z', 'L' and 'R'. The string length is not less than 1 and doesn't exceed 10^6.

Output

Write a required string to the output file.

Examples

stdin	stdout
abLcd	acdb
icpLLLLLacmRRRRRRRRRRRRc	acmicpc

Problem G. Friends of Friends

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	64 megabytes

Social networks are very popular now. They use different types of relationships to organize individual users in a network. In this problem friendship is used as a method to connect users. For each user you are given the list of his friends. Consider friendship as a symmetric relation, so if user a is a friend of user b then b is a friend of a.

A friend of a friend for a is such a user c that c is not a friend of a, but there is such b that b is a friend of a and c is a friend of b. Obviously $c \neq a$.

Your task is to find the list of friends of friends for the given user x.

Input

The first line of the input contains integer numbers N and x ($1 \le N \le 50$, $1 \le x \le N$), where N is the total number of users and x is user to be processed. Users in the input are specified by their numbers, integers between 1 and N inclusive. The following N lines describe friends list of each user. The i-th line contains integer d_i ($0 \le d_i \le 50$) — number of friends of the i-th user. After it there are d_i distinct integers between 1 and N — friends of the i-th user. The list doesn't contain i. It is guaranteed that if user a is a friend of user b then b is a friend of a.

Output

You should output the number of friends of friends of x in the first line. Second line should contain friends of friends of x printed in the increasing order.

Examples

stdin	stdout
4 2 1 2 2 1 3 2 4 2 1 3	1 4
4 1 3 4 3 2 3 1 3 4 3 1 2 4 3 1 2 3	0

Problem H. Berodoskar Development

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	64 megabytes

Not far away from Berland there is a small stony isle in the ocean. The name of the isle is Berodoskar and it is an ideal place for a military base. The only problem is lack of fresh water, but such a negligible problem will never stop Berl III, the King of Berland. He has stated that there would be exactly two water-storage reservoirs with desalinated water. The best way to organize reservoirs is to use existing craters located on the isle and connect them with the ocean by a water-pipe. And you are the person responsible for the water-storage system creation.

You have a map of Berodoskar on which the isle is represented as a rectangle divided into unit squares by horizontal and vertical lines. Each unit square is marked with either '`X`' character if it contains a part of crater or with '`.`' character if there is no crater on this square. If two squares marked with '`X`' have the common edge they are considered as parts of one crater. There are no squares marked with '`X`' connected with the ocean. The whole area around the square is the ocean.

The water-pipe you are to create should connect any two of craters with the ocean. It is allowed to connect more than two craters, but at least two craters should be connected. The water-pipe will consist of segments connected by edge and each segment will occupy the whole non-crater unit square. A pipe is connected with a crater if they have a common edge. You want to minimize number of squares involved into water-storage system creation. It is possible that two independent water-pipes are more efficient than one branched (see examples).

Input

The first line contains two integer numbers N and M — number of unit squares in vertical and horizontal directions ($3 \le N, M \le 15$). After that N lines follow describing the isle. Each line contains exactly M characters '`X`' or '`.`'. It is guaranteed that there are at least two craters reachable from the ocean.

Output

Output N lines with M characters in each — an isle map is in the same format as in the input, but with the water-pipe indicated. Squares with the water-pipe should be marked with '`*`' instead of '`.`'. If there are several solutions with the same minimal number of squares used, output any one of them.

Examples

stdin	stdout
5 7 ....X.... .....X. 	.... **X.... .....X* ....
10 13 ....XXXXX...... ..X..X....... ..X.X.X.X.... ..X...X...... ..XXXXX...... 	........*.... ....*.... ..XXXXX**.... ..X..X..*.... ..X.X.X.X.... ..X...X...... ..XXXXX...... ....

Note: in second example you are not allowed to connect a crater with the ocean via another crater.

Problem I. The last hour of the contest

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	64 megabytes

It's your lucky day - your team took part in the world finals programming contest. The competition has just finished and all competitors are looking forward to the results. Of course your team is not an exception.

According to the rules of the competition, duration of the contest is 5 hours and team standings are frozen during the last hour of the contest — from 241^{st} to 300^{th} minute. Due to this reason the actual results are unknown to the participants and are only announced at the awards ceremony. It is a long-term tradition though, to get a balloon at the world finals for every problem solved. This information can sometimes be of use during the competition.

While waiting for the awards ceremony, your team decided to evaluate chances to win and determine the place which it can take in the final standings. Here is the information that you have:

- Standings table at the end of the 240^{th} minute
- Total amount of balloons given to all teams within the last hour
- Exact rules of the competition

To solve this problem you should mind the following contest rules:

- The competition lasts for 300 minutes, minutes are numbered from 1 to the 300.
- Position of a team is defined by the amount of the problems solved and penalty time.
- The more problems are solved the higher place a team gets. If two or more teams have the same amount of problems solved, the sorting is done by the penalty time in ascending order (less penalty time — higher place). If two or more teams have the same amount of problems solved and the same penalty time, they share the same place. Rank of a team equals the number of teams performing better +1. If there are no other teams performing better, the team takes the 1^{st} place.
- Total penalty time is the sum of the penalty times consumed for each problem solved. The time consumed for a solved problem is the number of a minute (i. e. an integer between 1 and 300) when the correct submission was made plus 20 penalty minutes for every previously rejected run for that problem. There is no penalty time for a problem that is not solved.
- When solution for a particular problem is accepted, a team is not allowed to submit solutions of the same problem anymore.

Frozen standings are given to you in a text form and look as follows:

- The first line consists of "`Rank`", "`Team`", "`=`", and "`Penalty`" tokens followed by a list of problems identifiers. Every problem identifier is a capital Latin letter 'A'-'Z'. All problem identifiers are different.
- The second and the subsequent lines contain team standings itself. These lines are given to you. in non-descending order of team rank
- Every line contains a place of a team, a name of a team, the amount of problems solved and the result for every problem.

- Every result on a problem is
  - character "`.`", if the team didn't submit the problem;
  - "`-x`", if the problem was not accepted and the team made x unsuccessful submits;
  - "`+x`", if the problem was accepted and the team made x unsuccessful submits before the accepted run; x is always omitted if it equals to 0.

You are given the frozen standings, and your task is to calculate the minimum and the maximum place your team can get after the contest is over.

Input

Input file contains one or more set of input data.

First lines of each set contain the standings table at the end of the 240^{th} minute in the format described above. In each line the tokens are split with at least one whitespace character. The next line contains the amount of balloons given to all teams within the last hour of the contest (from 241^{st} to 300^{th} minutes). The last line of the set is the name of your team. All team names consist of Latin letters 'A'-'Z', 'a'-'z', numbers and whitespaces. Each team name contains no more than 100 characters, doesn't start/end with a whitespace, and contains not less then 1 non-whitespace character. All team names are unique within a single set of input data. The number of teams is between 1 and 100. The number of problems is between 1 and 26. The total number of rejected attempts in frozen standings table is not more than 1000, but the total number of rejected attempts in final standings is unlimited. The number of balloons is non-negative.

You may assume that the size of the input file is not more than 100 KB.

Output

For every input data set, please print two integer numbers in the line — the minimum and the maximum place your team can get after the contest is over.

Examples

stdin	stdout
Rank Team = Penalty X Y 1 Tarasov SU 3 2 33 + + 2 IMHO 1 1 20 + -1 3 Mozgow SU x 33 1 30 . + 3 MiTV 1 30 + -3 5 Opel SU 0 0 . . 2 MiTV Rank Team = Penalty A 1 aa 0 0 -1 1 ba 0 0 -8 2 ba	2 4 1 2

Problem J. Geologist Dubrovsky

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	64 megabytes

Geologist Dubrovsky travels a lot around the world and often faces different unusual phenomena. And now he found a unique place on the planet, where N rivers flow in parallel tight next to each other. The distance between each pair of neighbouring rivers can be neglected. Rivers flow from the south to the north. Geologist stays on the left bank of the most western river and wants to get to the right bank of the most eastern river.

The flow speed of i-th river is v_i meters per second and its width is w_i meters. Geologist Dubrovsky swims with the speed u meters per second in still water. If he swims across river, his real speed is a vector sum of his own speed vector and the speed vector of the river flow.

What is the maximal distance Dubrovsky can get from his original position by the time of sunset, if he has only t seconds left? Remember that his destination point is a point on the right bank of the easternmost river.

Input

The first line of the input contains three integer numbers N, u and t ($1 \le N \le 50$; $1 \le u, t \le 1000$). Each of the following N lines contains a pair of integers w_i, v_i ($1 \le w_i, v_i \le 1000$) describing corresponding parameters of the i-th river.

Output

To the first line of the output write the desired distance or -1 if geologist can't reach the right bank of the most eastern river in t seconds. The distance with a relative or absolute error of at most 10^{-6} will be considered correct. If solution exists the second line of the output should contain the sequence $t_1, t_2, \ldots, t_N$, where t_i is the time spent crossing the river i in the case of optimal track. The track is optimal if as a result Dubrovsky gets to the point on the right bank of the easternmost river, which is the farthest from his original position.

Examples

stdin	stdout
1 1 1 1 1	1.4142135624 1.0000000000
2 1 6 1 1 2 1	11.5911099155 2.0000000000 4.0000000000

Problem K. Terrorists in Berland

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	64 megabytes

Many countries dream of capturing great and glorious country of Berland. Berland is a big country consisting of N cities, some of them are connected by bidirectional roads. Each pair of cities is connected by no more then one road. A traveler can get from one city to any other city traveling by the roads only.

Invaders plan to use these roads during their operation, but Berland military forces are strong enough to destroy an enemy trying to move along the roads. The treacherous plan was developed to make the invasion easier: Berland should be divided into two parts. To make the plan more effective the decision was made to use terrorists from the Berland opposition. Several groups agreed to participate in the plan. The terrorists' resources are limited, so they can only destroy roads, not cities. The procedure of the road destruction is a costly operation. The security precautions on different roads can vary, so the cost of the road destruction can be different.

Invaders want to suddenly attack Berland without declaring the war and quickly capture one of the cities. Berland troops can't move through the captured city as they can't move along the destroyed roads.

Your task is to find cheapest plan of roads destruction, so that at least one city exists which can be captured and allow invaders to divide the country into two parts. Berland is considered divided, if two such cities exist, that there is no way along non-destroyed roads and through non-captured cities from one to another.

Input

The first line of the input contains two integer numbers N and M ($3 \le N \le 50$; $1 \le M \le 500$), where N is the number of cities and M is the number of roads. The following M lines contain the description of the roads given as three integer numbers a_i, b_i, w_i ($1 \le a_i < b_i \le N$; $1 \le w_i \le 10$). The cost of destruction of the the road from a_i to b_i is w_i.

Output

Write to the first line the total amount of money required to destroy roads in the optimal plan. Write to the second line K — number of roads to be destroyed according to the plan. Write to the third line numbers of roads to be destroyed divided by one space. The roads are numbered in order of appearance in the input. If there are several solutions, choose any of them.

Example

stdin	stdout
3 3 1 2 1 2 3 2 1 3 2	1 1 1
4 6 1 2 1 1 3 1 2 3 2 1 4 1 2 4 2 3 4 3	2 2 2 4