Task: BOW
Bowling

english

BOI 2015, day 1. Available memory: 256 MB. *30.04.2015*

Byteasar is a fan of both bowling and statistics. He has written down the results of a few past bowling games. Unfortunately, some characters in the notes are blurry, and thus unreadable. Byteasar asks you to write a program to calculate the number of distinct games which are consistent with his notes.

Rules of Bowling

A bowling game consists of n *frames*: $n-1$ *simple* frames and one *final* frame. In a typical game $n = 10$. At the beginning of each frame 10 *pins* are put standing upright at the end of a lane and a player gets no more than two (or three for the final frame) attempts (*shots*) to throw a bowling ball down the lane to try to knock down as many pins as possible. Each frame is denoted by two (for a simple frame) or three (for the final frame) characters.

For each shot the player receives as *basic points* the total number of pins knocked down in this shot. The player's *basic points* in each frame are the sum of basic points of all the shots in this frame. If all 10 pins are knocked down in a simple frame (and therefore 10 basic points are earned), the player gets additional *bonus points*.

For a simple frame the rules are the following:

- If a player knocks down all 10 pins in the first shot of a frame, she gets a *strike* and the frame ends. As bonus points she gets the sum of basic points of her next two shots. A strike is denoted as "`x-`".
- If a player knocks down all 10 pins using both shots of a frame, she gets a *spare*. As bonus points she gets the basic points of her next shot. A spare is denoted as "A/", where A is a digit describing the number of pins knocked down in the first shot of the frame.
- If 9 or fewer pins are knocked down after both shots, the player gets just basic points and such a frame is denoted as "AB", where A is the one-digit number of pins knocked down in the first shot, and B is the one-digit number of pins knocked down in the second shot ($A + B < 10$).

Note that bonus points are included to the score of a frame in which the strike or the spare was obtained, regardless of the fact that the exact number of bonus points depends on future shots in next frames.

For the final frame the rules are the following:

- Initially the player receives two shots in this frame. If 9 or fewer pins are knocked down in the two shots, the frame ends. Otherwise (if the first two shots are a spare or the first shot is a strike), the player receives a third shot in the frame. Whenever the player knocks down all the pins in any of the three shots, the pins are reset to the initial configuration for the next shot. The score of the final frame is the total number of pins knocked down (note that no bonus points are earned due to strikes and spares).
- Overall there are seven possible configurations of the final frame with the following outcomes (A and B stand for one-digit numbers):

Denotation	Description	Frame Score
“`xxx`”	three consecutive strikes	30
“`xx`A”	two consecutive strikes and a shot with A pins knocked down	$20 + A$
“`x`A`/`”	a strike and a spare with A pins knocked down on its first shot	20
“`x`AB”	a strike and two following shots with A and B pins knocked down respectively ($A + B < 10$)	$10 + A + B$
“A`/x`”	a spare with A pins knocked down on the first shot and a strike	20
“A`/`B”	a spare with A pins knocked down on the first shot and B pins knocked down in the last shot	$10 + B$
“AB`-`”	two shots with A and B pins knocked down respectively ($A+B < 10$)	$A + B$

Each game is described as a sequence of $2n + 1$ characters. At the end of the game the total number of points after each frame may be calculated. For example, for a game of $n = 10$ frames described as “`08x-7/2/x-x-23441/0/x`”, the player’s points after respective frames were as follows:

Frame	Denotation	Basic Points	Bonus Points	Frame Score	Total
1	“`08`”	$0 + 8$	—	8	8
2	“`x-`”	10	$7 + 3$	20	28
3	“`7/`”	$7 + 3$	2	12	40
4	“`2/`”	$2 + 8$	10	20	60
5	“`x-`”	10	$10 + 2$	22	82
6	“`x-`”	10	$2 + 3$	15	97
7	“`23`”	$2 + 3$	—	5	102
8	“`44`”	$4 + 4$	—	8	110
9	“`1/`”	$1 + 9$	0	10	120
final	“`0/x`”	$0 + 10 + 10$	—	20	140

Input

The first line of input contains one integer q ($1 \le q \le 25$), specifying the number of test cases to consider. The following $3q$ lines of input contain descriptions of test cases. Each test case is described by three lines of input.

The first line of a test case description contains one integer n ($2 \le n \le 10$), specifying the number of frames. The second line contains a sequence of $2n + 1$ characters which denotes the game description from Byteasar’s notes. Blurry characters are replaced by “`?`” characters. The third line contains n integers, the total number of points after each frame, separated by spaces. In each number either all digits are readable, or all digits are blurry. Numbers in which all digits are blurry are replaced by “`-1`”.

Output

Your program should output q lines, one line per each test case in the same order as in the input.

For each test case your program should write one integer: the number of possible distinct games corresponding to the test case. Two games are considered different if and only if they differ in at least one shot, that is, their $(2n+1)$-character game descriptions are different. You can assume that there is at least one game consistent with each test case in the input. You can assume that the result fits into 64-bit signed integer type.

Examples

For the input data:

```
2
10
08x-7/2/x?x-23??1/???
8 -1 40 60 82 97 102 110 120 140
5
x-x-23?/00-
22 37 42 52 52
```

a correct result is:

```
9
10
```

Explanation to the examples: In the first case, in frame 5 after the character "`x`" the only possible character is "`-`". In frame 8 the player got 8 points in total. Thus there are 9 possibilities how this sum could have been obtained: $0+8, 1+7, \ldots, 8+0$. There were no bonus points in frame 9. Therefore, there were no points on the first shot of the final frame. To obtain 20 points in the last two shots, the only possibility is a spare with a following strike in the last shot of the frame. Therefore there are 9 different valid games which correspond to this input data.

In the second case, any character from `0` to `9` is consistent with the input data.

Additional sample test: In the contest system we provide you with an additional sample test with multiple test cases with $n = 2$.

Grading

Subtask	Conditions (in each test case)	Points
1	at most six "`?`" characters in the input sequence	16
2	the result is at most 10^9	17
3	no game whose description contains any characters "`x`" or "`/`" is consistent with the input data	26
4	the input sequence ends with "`00-`" (that is, the player obtained 0 points in the last frame) and last $\min(3, n)$ frame scores provided in the third line of input data are all "`-1`"	23
5	no additional constraints	18

Task: EDI
Editor

english

BOI 2015, day 1. Available memory: 512 MB. *30.04.2015*

Byteasar is a programmer who works on a revolutionary text editor. In the editor there are two types of operations: one type allows to *edit text* in the editor, and the other type allows to *undo* previously performed operations. One of the innovative features of this editor is a *multilevel undo operation*. It works as follows. We say that a text editing operation is an operation of *level 0*. An *undo operation of level i* (for $i = 1, 2, \ldots$) undoes the last operation of level at most $i - 1$ which is not undone. For instance, an undo operation of level 1 can undo only editing operations, and an undo operation of level 2 can undo editing operations as well as undo operations of level 1 (but no undo operations of greater levels).

More formally, each of the already performed operations can be in two states: `active` or `undone`. Let X be one of the operations. Just after performing the operation X, it is in the state `active`. If X is an undo operation of level i, we find the most recent operation in state `active` of level at most $i - 1$ (denote it by X_1) and change the state of the operation X_1 to `undone`. If X_1 is also an undo operation, we must change to `active` the state of the operation which X_1 had undone (say X_2). We continue in the same manner: whenever the state of an undo operation X_j which had previously undone some operation X_{j+1} changes, we must also change the state of the operation X_{j+1} (which, of course, may result in changing states of further operations). The whole chain of state modifications finishes when an editing operation is reached.

For simplicity, the current contents of text in the editor will be specified by a single integer s, called the *editor state* (equal to 0 at the beginning). Each editing operation specifies the editor state that it produces. The editor state depends on the last editing operation in the state `active`. Help Byteasar and write a program which keeps track of the editor state.

Let us see this in action: the following table shows some operations performed by Byteasar and the editor state after performing each of them. The symbol $\mathtt{E}_s$ denotes an editing operation which changes the editor state to s, whereas the symbol $\mathtt{U}_i$ denotes an undo operation of level i.

Operation		$\mathtt{E}_1$	$\mathtt{E}_2$	$\mathtt{E}_5$	$\mathtt{U}_1$	$\mathtt{U}_1$	$\mathtt{U}_3$	$\mathtt{E}_4$	$\mathtt{U}_2$	$\mathtt{U}_1$	$\mathtt{U}_1$	$\mathtt{E}_1$
Editor state	0	1	2	5	2	1	2	4	2	1	0	1

First, Byteasar performed three editing operations. The editor state changed from 0 to 1, then to 2, and finally to 5. Next, he performed two undo operations of level 1, which undid the operations $\mathtt{E}_5$ and $\mathtt{E}_2$ (changing their state to `undone`). Thus the editor state was restored to 1. The following undo operation of level 3 undid the last operation $\mathtt{U}_1$ (changing its state to `undone`), consequently restoring the operation $\mathtt{E}_2$ (changing its state back to `active`). As a result the editor state changed once again to 2. Operation $\mathtt{U}_2$ undid the operation $\mathtt{E}_4$, operation $\mathtt{U}_1$ once again undid the restored operation $\mathtt{E}_2$, the last operation $\mathtt{U}_1$ undid the operation $\mathtt{E}_1$, and the final operation is $\mathtt{E}_1$.

Input

The first line of the input contains a positive integer n, specifying the number of operations performed by Byteasar. The next n lines contain descriptions of operations, one per line, each being an integer a_i ($-n \le a_i \le n$, $a_i \neq 0$). If $a_i > 0$, then it specifies an editing operation which modifies the editor state to a_i. If $a_i < 0$, then it specifies an undo operation of level $-a_i$. You can assume that for every undo operation there will be some operation in the state `active` of smaller level to undo.

Output

Your program should output n lines. The i-th line should contain one integer specifying the editor state after performing the first i operations from the input.

Examples

For the input data:	a correct result is:
`11`	`1`
`1`	`2`
`2`	`5`
`5`	`2`
`-1`	`1`
`-1`	`2`
`-3`	`4`
`4`	`2`
`-2`	`1`
`-1`	`0`
`-1`	`1`
`1`	

Grading

Subtask	Conditions	Points
1	$n \leq 5000$	20
2	$n \leq 300\,000$ and there are only operations $\mathtt{E}_i$ and $\mathtt{U}_1$	15
3	$n \leq 300\,000$ and only the last number in the sequence is graded (however, the first $n-1$ numbers must be integers ranging from 0 to n)	28
4	$n \leq 300\,000$	37

Task: NET
Network

english

BOI 2015, day 1. Available memory: 256 MB. *30.04.2015*

The government of Byteland has decided that it is time to connect their little country to the Internet, so that all citizens can participate in programming competitions and watch videos of cute cats. When it was time to build the network backbone of the country, they assigned the company Internet Optimists Inc. with connecting all the n computers of Byteland. The connections were made as direct links between pairs of computers in such a way that any pair of computers are connected by a sequence of links.

Byteland is not a rich country by any means, so to minimize costs the network topology was built in the form of a *tree* (i.e. there are exactly $n-1$ direct links between computers). Far too late, it was realised that this solution suffers from a serious drawback. If just a single link is broken, the computers of Byteland will be partitioned so that some computers cannot communicate with each other!

To improve the reliability of Byteland's network, it was decided that it should at least tolerate if a single link is broken. Your task is to help Internet Optimists Inc. to improve the network in a cheapest way. Given the network topology of Byteland (i.e. which $n-1$ pairs of computers are connected by direct links), find the minimum number of links that need to be added so that the network will still be connected if any single link is broken.

Input

The first line of input contains a positive integer n ($n \geq 3$), the number of computers in Byteland. For simplicity, all computers are numbered from 1 to n. Each of the following $n-1$ lines contains a pair of integers a and b ($1 \leq a, b \leq n$, $a \neq b$) that describes a direct link between computers a and b.

Output

In the first line of output your program should write an integer k, the minimal number of links that should be added to the network. In each of the following k lines your program should write a pair of integers a and b ($1 \leq a, b \leq n$, $a \neq b$) that denote the numbers of computers that should be connected by a link. The links can be written in any order. If there is more than one solution, your program should output any one of them.

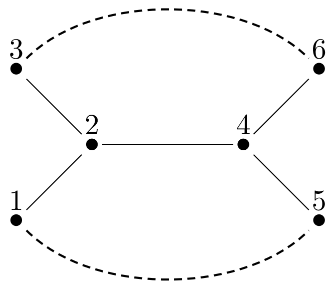
Examples

For the input data:

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

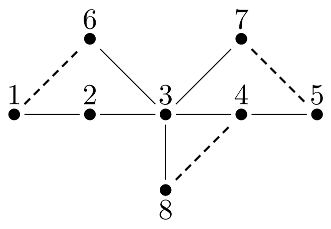
a correct result is:

```
2
1 5
3 6
```



For the input data:

```
8
1 2
2 3
3 4
4 5
3 6
3 7
3 8
```



a correct result is:

```
3
1 6
5 7
8 4
```

Grading

Subtask	Conditions	Points
1	$n \le 10$	18
2	$n \le 2000$	45
3	$n \le 500\,000$	37