Problem A. Asynchronous Processor

Time limit:	2 seconds
Memory limit:	1024 megabytes

You are given a program consisting of n instructions executed by a processor with a single integer register A, initially equal to 0. Each instruction is one of two types:

- "`+ v`" — performs $A := A + v$;
- "`= v`" — performs $A := v$.

The instructions in the program are numbered from 1 to n. Each instruction i initially has timestamp i.

Some instructions are marked as *asynchronous*. If instruction i is asynchronous, its timestamp can be changed to any **real** number greater than i.

After all timestamp adjustments, all timestamps must be distinct. The processor then executes the instructions in order of increasing timestamp.

Determine the number of distinct final values of A that can be obtained after all instructions have been executed, considering all possible choices of asynchronous instruction timestamps.

Input

The first line contains an integer n, denoting the number of instructions in the program ($1 \le n \le 2000$).

The i-th of the following n lines describes instruction i and contains three tokens. The first token is either '`+`' or '`=`', denoting the type of the instruction. The second token is an integer v, denoting the argument of the instruction ($1 \le v \le 500$). Finally, the third token is either "`async`" if the instruction is marked as asynchronous, or "`sync`" otherwise.

Output

Print the number of distinct final values A can take after executing the program.

Examples

standard input	standard output
3 + 1 sync = 2 async + 3 async	2
10 = 7 async + 3 async + 5 sync + 3 async = 1 sync + 9 async + 10 async + 1 sync + 3 async + 4 sync	30

Note

In the first test, the program execution starts with instruction 1 setting A to 1. Then, instructions 2 and 3 are executed in one of the two orders:

- if "`= 2`" is executed before "`+ 3`", A will be equal to 5;
- if "`+ 3`" is executed before "`= 2`", A will be equal to 2.

Thus, there are two possible values of A at the end: 5 and 2.

Problem B. Bounding Boxes

Time limit:	2 seconds
Memory limit:	1024 megabytes

ICPC is considering sending out contest souvenirs via a delivery service. The delivery service provides multiple types of packing boxes, each shaped as a rectangular parallelepiped.

Unfortunately, there is no way to know which packing box type will be available on the shipping day, so ICPC needs to choose a souvenir box size that fits inside every packing box.

According to the shipment rules, the souvenir box must also be a rectangular parallelepiped. When placed inside a packing box, the souvenir box may be rotated, but its sides must remain parallel to the sides of the packing box. Extra space is not an issue, as it will be filled with plastic wrap.

Help ICPC determine the maximum possible volume of a souvenir box that fits inside all of the packing boxes.

Input

The first line contains a single integer n, denoting the number of packing box types provided by the delivery service ($1 \le n \le 1000$).

The i-th of the following n lines contains three integers w_i, h_i, and d_i, denoting the width, height, and depth of the i-th packing box ($1 \le w_i, h_i, d_i \le 1000$).

Output

Print the largest possible volume of a souvenir box that fits inside all of the packing boxes. Remember that the souvenir box can be rotated, as long as its sides remain parallel to the sides of the packing box.

Example

standard input	standard output

3
6 5 6
2 10 10
3 8 4
	48

Note

In the example, the largest souvenir box that fits in each of the packing boxes has dimensions $2 \times 4 \times 6$. It fits in the first two packing boxes without rotation, and it can be rotated to become $2 \times 6 \times 4$ to fit in the third packing box.

Problem C. Compact Encoding

Time limit:	2 seconds
Memory limit:	1024 megabytes

Binary formats often use compact representations for integers. Consider writing a 32-bit unsigned integer to a file: you must always reserve 4 bytes to represent it (remember, 8 bits make one byte). However, in many real-life applications, integer values tend to be small. Writing these small values using a fixed 4-byte representation results in files that are mostly filled with zero bytes.

To make the representation more compact, we introduce the following encoding scheme. Each value is represented as a sequence of bytes $b_1, b_2, \ldots, b_k$, where each b_i is an integer between 0 and 255, inclusive. The most significant bit of each byte serves as a continuation flag, and the lower 7 bits carry the actual data. If the continuation flag is 1, more bytes follow; for the last byte, it is 0. The representation is *big-endian*, meaning that b_1 contains the most significant bits of the encoded value.

For example, here's how to find the compact representation of $n = 112025$. First, we find its binary representation:

$$112025 = 11011010110011001_2$$

Next, we split it into 7-bit chunks, padding with zeros on the left if necessary:

$$0000110 \ / \ 1101011 \ / \ 0011001$$

The first two chunks have a following chunk, so the corresponding bytes have their most significant bit set to 1. The last chunk has no following chunk, so its most significant bit is 0. This gives us:

$$b_1 = 10000110_2 = 134$$
$$b_2 = 11101011_2 = 235$$
$$b_3 = 00011001_2 = 25$$

You are given an integer n, and your task is to find its compact representation.

Input

The only line contains a single integer n ($0 \le n \le 2^{31} - 1$).

Output

Print a sequence of integers between 0 and 255, inclusive, representing the compact encoding of n. The encoding must not contain leading bytes with zero data bits: that is, it may not start with 128.

Examples

standard input	standard output
112025	134 235 25
128	129 0
0	0
42	42
16384	129 128 0
2147483647	135 255 255 255 127

Problem D. Defense Distance

Time limit:	2 seconds
Memory limit:	1024 megabytes

The NWRRC security server has a final access check for teams that try to submit their solutions to the secret hidden problem.

To pass the check, the team must enter three passwords s, t, and u that the system will accept. Each password must be a non-empty string consisting of at most 5000 lowercase English letters.

The rules of the server are public:

- The distance between s and t should be equal to a.
- The distance between s and u should be equal to b.
- The distance between t and u should be equal to c.

The *distance* between two strings s_1 and s_2 is the minimum number of single-character operations (insert one character, remove one character, or replace one character) needed to convert string s_1 into string s_2. This metric is also known as the Levenshtein distance.

The server gives access to the hidden problem if and only if all described conditions are satisfied. Your goal is to construct a triple of passwords to unlock the hidden problem or determine that it is impossible.

Input

The only line contains three integers a, b, and c, denoting the required distances between each pair of passwords ($0 \le a, b, c \le 1000$).

Output

If there are no three passwords with the required properties, print "`No`" in the only line.

Otherwise, print "`Yes`" in the first line. Then print passwords s, t, and u in the following three lines. Each password should consist of at least 1 and at most 5000 lowercase English letters.

If there are multiple triples of passwords that meet the requirements, print any of them.

Examples

standard input	standard output

`4 3 5`	`Yes`
	`icpc`
	`nwrrc`
	`itmo`

`2 2 2`	`Yes`
	`aa`
	`bb`
	`cc`

`0 0 1`	`No`

Note

In the first test case:

- The distance between "`icpc`" and "`nwrrc`" is 4: "`icpc`" $\to$ "`irpc`" $\to$ "`irrc`" $\to$ "`nrrc`" $\to$ "`nwrrc`".
- The distance between "`icpc`" and "`itmo`" is 3: "`icpc`" $\to$ "`itpc`" $\to$ "`itpo`" $\to$ "`itmo`".
- The distance between "`nwrrc`" and "`itmo`" is 5: "`nwrrc`" $\to$ "`wrrc`" $\to$ "`wrro`" $\to$ "`irro`" $\to$ "`itro`" $\to$ "`itmo`".

In the second test case, the distance between each pair of passwords is 2.

In the third test case, it can be shown that there are no three passwords with the required properties.

Problem E. Eight-Connected Figures

Time limit:	4 seconds
Memory limit:	1024 megabytes

This is an interactive problem.

An infinite square grid is hidden from you. Every cell is identified by a pair of integers (x, y) and is **randomly** colored either black or white with 50% probability for each color, independently of other cells.

Two cells are considered *adjacent* if they share an edge or a corner. Thus, every cell (x, y) has 8 adjacent cells: $(x-1, y-1)$, $(x-1, y)$, $(x-1, y+1)$, $(x, y-1)$, $(x, y+1)$, $(x+1, y-1)$, $(x+1, y)$, and $(x+1, y+1)$.

A set of cells S is called *8-connected* if for any two cells in S, there exists a path between them using only cells from S, where consecutive cells in the path are adjacent.

In one query, you can learn the color of any cell on the grid. Your task is to find an 8-connected set of n cells such that all cells in the set have the same color.

You need to solve t test cases. In each test case, the grid is colored randomly and independently of the other test cases.

You are allowed to make at most 30 000 queries **in total** over all test cases.

Input

The first line contains two integers t and n, denoting the number of test cases and the required size of the 8-connected set ($1 \le t \le 50$; $2 \le n \le 300$).

Interaction Protocol

In each test case, you can make zero or more queries to learn the colors of grid cells.

To make a query, print a line:

- `?` x y

where (x, y) are the coordinates of the requested cell ($-10^9 \le x, y \le 10^9$). After that, read a line containing one letter: '`B`' if the cell (x, y) is black, or '`W`' if the cell (x, y) is white.

Once you are ready to present an 8-connected set of n cells of the same color, print a line:

- `!` c x_1 y_1 x_2 y_2 $\ldots$ x_n y_n

where c is a letter denoting the color of the cells in the set ('`B`' for black and '`W`' for white), and $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ are the n distinct cells in the set ($-10^9 \le x_i, y_i \le 10^9$). The interactor does not print anything in response to this line.

After printing the set, proceed to the next test case, or terminate the program if this was the last one.

You are allowed to make at most 30 000 queries **in total** over all test cases (not including the lines that print the sets). If you exceed this limit, the interactor will print 0 instead of its usual response, and your program should terminate immediately to guarantee the "Wrong Answer" verdict.

The interactor is not adaptive: all random grids used in the tests have been pre-generated and remain the same across all submissions.

Example

standard input	standard output
2 5	
	? 1 1
W	
	? 1 2
W	
	? 1 3
B	
	? 2 1
B	
	? 2 2
B	
	? 2 3
W	
	? 3 1
B	
	? 3 2
B	
	? 3 3
B	
	! B 2 2 1 3 3 3 2 1 3 2
	? 1 1
B	
	? 1 2
W	
	? 1 3
W	
	? 2 1
B	
	? 2 2
B	
	? 2 3
W	
	? 3 1
W	
	? 3 2
W	
	? 3 3
B	
	! W 1 2 3 2 1 3 2 3 3 1

Note

In the example, the queries and the responses are separated by empty lines for clarity. In the actual interaction between your program and the interactor, there will be no empty lines.

Your solution will be evaluated on at most 60 test files.

Problem F. Faulty Fraction

Time limit:	2 seconds
Memory limit:	1024 megabytes

Felix is studying basic arithmetic at school. Today he learned division. As a final task, he calculated the result of the division of a positive integer a by a positive integer b. The result was a positive integer c, since a was divisible by b. Felix wrote $a \div b = c$ in his notebook and went outside to play football.

His little sister Fiona had been watching his studies with great interest. When Felix left, she decided to play a little trick on him: she took his notebook and erased the '$\div$' sign from the equation. As a result, the left-hand side of the equation became a single string of digits s.

Once Felix came back, he saw $s = c$ in his notebook. Unfortunately, he forgot the original values of a and b. Now he needs to split s back into two parts using the '$\div$' sign to restore a correct division equation.

Help Felix find positive integers a and b such that s is the concatenation of the decimal representations of a and b, and $a \div b = c$.

Input

The only line contains a string of digits s and an integer c. Both s and c consist of at least 1 and at most 10^5 digits and do not have leading zeros.

It is guaranteed that s is a concatenation of two positive integers a and b written without leading zeros such that $a \div b = c$.

Output

Print two positive integers a and b without leading zeros such that s is the concatenation of a and b, and $a \div b = c$. If there are multiple answers, print any of them.

Examples

standard input	standard output
`42 2`	`4 2`
`2025225 9`	`2025 225`
`239239239 1001`	`239239 239`

Problem G. Games of Chess

Time limit:	2 seconds
Memory limit:	1024 megabytes

There are n friends and n houses in Chess City, both numbered from 1 to n, where friend i lives in house i. The houses are connected by m bidirectional roads, forming a connected network.

Chess City also has n virtual chess clubs, numbered from 1 to n. Each friend must choose exactly one club to join. These choices do not need to be distinct, so some clubs might not have any members.

When friend i hosts a chess party, it is attended by every friend who belongs to the same club as friend i and lives in a house directly connected to house i by a road. The party is considered *successful* if the total number of attendees, including friend i, is even; in this case, they can all play chess simultaneously.

Choose a club for each friend so that every friend's party is successful, or report that it is impossible.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n and m, denoting the number of houses and the number of roads ($2 \le n \le 10^5$; $n - 1 \le m \le 2 \cdot 10^5$).

Each of the following m lines contains two integers u and v, describing a bidirectional road between houses u and v ($1 \le u, v \le n$; $u \ne v$). No two houses are connected by more than one road. It is possible to get from any house to any other one using the roads.

It is guaranteed that the sum of n over all test cases does not exceed 10^5, and the sum of m over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer -1 if it is impossible to assign a chess club to each friend so that every friend's party is successful.

Otherwise, print n integers $c_1, c_2, \ldots, c_n$, where c_i is the number of the chess club that friend i should join ($1 \le c_i \le n$). If there are multiple answers, print any of them.

Example

standard input	standard output

3
2 1
1 2
3 3
1 2
2 3
3 1
8 10
1 2
1 4
1 7
5 2
5 4
5 7
5 3
2 6
2 8
6 8
	1 1
	-1
	3 3 6 6 6 2 6 2

Problem H. High Score

Time limit:	2 seconds
Memory limit:	1024 megabytes

Hermione loves to play the following computer game, in which the player controls an unordered multiset of integers. Initially, the multiset is empty and the player's score is 0. At any moment in the game, the multiset contains at most k integers (not necessarily distinct). In one turn, the player can choose one of the following moves:

- **Insert**. Choose an integer 2 or 4 and insert it into the multiset. This move does not change the score, and it is only allowed if the size of the multiset before the move is strictly less than k.
- **Merge**. Choose an integer x such that the multiset contains at least two copies of x. Remove two copies of x from the multiset, and insert one copy of $2x$ into the multiset. This move adds the value $2x$ to the player's score.

The player can stop the game after any turn, at which point the player's score becomes final.

Hermione has been on vacation for the whole summer, and she hasn't played the game in a while. Today, she opened the game on her laptop and saw the leaderboard with the highest scores she's ever had: $h_1, h_2, \ldots, h_n$ in some order. Now she is curious how she managed to achieve those scores.

For each h_i, find any multiset of integers that Hermione could have at the end of the game if her final score was h_i, or determine that it is impossible to achieve such a score.

Input

The first line contains two integers n and k, denoting the number of scores on the leaderboard and the maximum size of the multiset ($1 \le n \le 10^4$; $2 \le k \le 16$).

Each of the next n lines contains a single integer h_i, denoting a score on the leaderboard ($1 \le h_i \le 10^9$).

Output

For each score h_i, print the final size of the multiset s, followed by s integers describing the contents of the multiset in any order ($0 \le s \le k$). It must be possible to achieve the score h_i with this multiset at the end of the game. If there are multiple answers, print any of them.

If it is impossible to have the score h_i at the end of the game, print a single integer -1 instead.

Examples

standard input	standard output
1 2 12	1 8

4 5
4
12
10
20
	2 4 2
	5 8 4 2 2 4
	-1
	3 2 4 8

1 16 19956	1 2048

Note

A possible sequence of moves for the first test is shown below:

$$\{\} \xrightarrow{\texttt{insert 2}} \{2\} \xrightarrow{\texttt{insert 2}} \{2,2\} \xrightarrow[\texttt{score += 4}]{\texttt{merge, x = 2}} \{4\} \xrightarrow{\texttt{insert 4}} \{4,4\} \xrightarrow[\texttt{score += 8}]{\texttt{merge, x = 4}} \{8\}$$

Problem I. Infection Investigation

Time limit:	3 seconds
Memory limit:	1024 megabytes

Isaac is a biologist who specializes in diagnosing viral diseases. The virus modifies the host genome (a sequence of genes) by altering it to suit its own needs. Isaac is writing a paper investigating viral infection in genomes. He has some samples and asks you to help analyze them.

For simplicity, we will assume that the viral genome consists of genes $1, 2, \ldots, n$ in this order. The host genome is a permutation of genes $1, 2, \ldots n$: it consists of genes $a_1, a_2, \ldots, a_n$ in this order.

Consider a genomic segment $[l; r]$ consisting of genes $a_l, a_{l+1}, \ldots, a_r$. The infection level of this segment is measured as the length of the longest subsequence of genes shared with the viral genome. Formally, the infection level is the maximum k such that there exist $l \le i_1 < i_2 < \cdots < i_k \le r$ for which the inequalities $a_{i_1} < a_{i_2} < \cdots < a_{i_k}$ hold.

To analyze the genome, Isaac needs to estimate the infection levels of q genomic segments. To secure the funding, Isaac only needs approximate results: an error factor of up to 1.5 is allowed.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n and q, denoting the host genome length and the number of genomic segments Isaac is interested in ($1 \le n, q \le 2 \cdot 10^5$).

The second line contains n distinct integers $a_1, a_2, \ldots, a_n$, describing the host genome ($1 \le a_i \le n$).

Each of the following q lines contains two integers l and r, denoting the boundaries of a genomic segment for which the infection level should be estimated ($1 \le l \le r \le n$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$, and the sum of q over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print q positive integers, denoting the infection levels of the corresponding genomic segments.

For each genomic segment, let your answer be x and let the true answer be y. Your answer will be considered correct if it differs from the true answer by a factor of at most 1.5, that is, if $\max\left(\frac{x}{y}, \frac{y}{x}\right) \le 1.5$.

Example

standard input	standard output

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

Problem J. Judging Problem

Time limit:	2 seconds
Memory limit:	1024 megabytes

The judges of NWRRC came up with n problems on a similar topic and decided to use them for n consecutive years, one problem per year. The only question was: in what order should they be used?

Each problem's name consists of two words. Let's call two names *similar* if either their first words are the same or their second words are the same. For example, `eight shaped` and `eight connected` are similar, while `hello world` and `world hello` are not.

The judges decided to implement the following rule: in the first year, they chose a problem arbitrarily. In every subsequent year, if there was a problem with a name similar to the previous year's problem that was still unused, they chose one of such problems; otherwise, they chose any unused problem.

You are given the names of the problems in chronological order of their use. Determine whether the judges correctly followed the rule above, or if they made a mistake.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n, denoting the number of problems ($2 \le n \le 10^5$).

The i-th of the following n lines contains the name of the i-th problem in chronological order: two words consisting of at least 1 and at most 10 lowercase English letters each. All problem names are distinct.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, print "`Yes`" if the judges followed the rule correctly, and "`No`" otherwise.

Example

standard input	standard output

```
3
4
k shaped
h shaped
eight shaped
eight connected
3
k shaped
eight connected
eight shaped
4
judging problem
judging logic
binary problem
logic problem
```
	```
	Yes
	No
	Yes
	```

Note

In the first test case, each subsequent problem name is similar to the previous one.

In the second test case, the judges should have chosen "`eight shaped`" for the second year.

In the third test case, neither "`binary problem`" nor "`logic problem`" is similar to "`judging logic`"; the judges could have chosen either of those problems for the third year.

Problem K. Keys and Grates

Time limit:	2 seconds
Memory limit:	1024 megabytes

Katniss is in a very long straight tunnel and wants to get out of it. She can move along the tunnel in both directions. There is exactly one hatch in the tunnel, and if she reaches it, she can exit.

Unfortunately, it's not that simple. There are grates blocking the entire width of the tunnel at n locations. The grates are locked, and Katniss cannot pass through a grate until she unlocks it. Fortunately, there are n keys located at n points along the tunnel. Each grate can be unlocked by exactly one of these n keys. Once a grate is unlocked, Katniss can freely and repeatedly pass through it. She can pick up and hold an unlimited number of keys.

Katniss knows her own location, as well as the locations of all n keys, all n grates, and the hatch. She also knows which key unlocks which grate. Determine the minimum distance she must travel to escape.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5 \cdot 10^4$). The description of the test cases follows.

The first line of each test case contains three integers n, s, and h, denoting the number of grates, the initial coordinate of Katniss, and the coordinate of the hatch ($1 \le n \le 2 \cdot 10^5$; $-10^6 \le s, h \le 10^6$).

The i-th of the following n lines contains two integers k_i and g_i, denoting the coordinate of the i-th key and the coordinate of the grate that can be unlocked with this key ($-10^6 \le k_i, g_i \le 10^6$).

All $2n + 2$ integers s, h, k_i, and ℓ_i are distinct.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print the minimum possible length of Katniss' escape route. If it is impossible to escape, print -1 instead.

Example

standard input	standard output
3	32
5 5 13	-1
3 7	7
8 14	
-1 11	
9 1	
2 10	
4 40 0	
20 80	
50 30	
70 60	
90 10	
1 -1 6	
11 8	

Note

In the first test case, one of the shortest escape routes goes as follows: 5 (initial) $\to$ 3 (pick up key 1) $\to$ 7 (unlock grate 1) $\to$ 9 (pick up key 4) $\to$ 1 (unlock grate 4) $\to$ -1 (pick up key 3) $\to$ 2 (pick up key 5) $\to$ 10 (unlock grate 5) $\to$ 11 (unlock grate 3) $\to$ 13 (hatch).

Problem L. Lucky Number Theory

Time limit:	4 seconds
Memory limit:	1024 megabytes

Lucy is a frequent arcade visitor. All machines at the arcade give out tickets to exchange for prizes! Lucy's favorite machine works as follows.

The machine only has two buttons: "Roll" and "Withdraw". Whenever Lucy presses "Roll", the machine increases the counter on the screen by a randomly generated **real** number between 0 and d. At any moment, she can press "Withdraw" to get the number of tickets equal to the counter, which gets reset. In case it's not an integer, the machine generously rounds the counter up before handing out the tickets.

More formally, the machine stores a real number S, initially equal to 0. On each "Roll" press, the machine generates Δ — a random **real** number picked uniformly from the interval $(0, d)$. Then, S increases by the value of Δ. When the "Withdraw" button is pressed, the machine rounds S up, giving the player $\lceil S \rceil$ tickets, and then resets S back to zero. Lucy can see the value of S on the screen at any moment with as much precision as she wants, and she can use it to decide whether to roll or withdraw.

Lucy has enough arcade tokens to press "Roll" n times, and "Withdraw" k times. Find a strategy that maximizes the expected number of tickets Lucy can get, and print this maximum expected number.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 2000$). The description of the test cases follows.

The only line of each test case contains three integers n, k, and d, denoting the number of rolls, the number of withdrawals, and the upper bound on roll values ($1 \le k \le n \le 2000$; $1 \le d \le 2000$).

Output

For each test case, print the maximum expected number of tickets Lucy can get, as a floating point number. Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-6}.

Example

standard input	standard output

3
3 2 1
5 5 3
7 1 10
	2.6250000000
	10.0000000000
	35.5000000000

Note

In the first test case, with $n = 3$ rolls, $k = 2$ withdrawals, and $d = 1$, the optimal strategy is as follows:

Lucy starts with a roll. Depending on the number S, there are two possibilities:

- The number is less than $\frac{1}{2}$. In this case, Lucy should withdraw, then roll two more times, and withdraw at the end. The expected number of tickets in this case is $1 + \frac{3}{2} = 2.5$ (1 ticket for the first withdrawal, and $\frac{3}{2}$ tickets on average after two rolls).
- The number is at least $\frac{1}{2}$. In this case, it's optimal to roll again, withdraw, then roll for the last time, and withdraw at the end. For the first withdrawal, she will get only one ticket with probability $\frac{1}{4}$ (the probability that the sum of the first two rolls is at most 1, given that the first roll was over $\frac{1}{2}$), and two tickets with probability $\frac{3}{4}$. The expected number of tickets is $1 + 2 \cdot \frac{3}{4} + 1 \cdot \frac{1}{4} = 2.75$.

Each case happens with probability $\frac{1}{2}$, so the total expected value for this strategy is $\frac{1}{2}(2.5+2.75) = 2.625$.

In the second test case, Lucy can withdraw after every roll, each time getting 2 tickets on average.

In the third test case, Lucy can only withdraw once, and she should do it after all 7 rolls.