IEEEXTREME PROGRAMMING COMPETITION 16.0

Problems 22-10-22

Array 3

Gaius Letters 4

Pizza Cutter 6

ASCII Dancer 8

Pipeline 11

ANDman 16

Dream Team 18

Palindrome Cutting 20

What Language Am I Speaking? 21

Scheduler 24

Scheduler Redux 25

Invisible Ghosts 26

My Treat 30

Book Cipher 32

Combo Locks 35

Travel Service 36

Outpost 38

Poker Tournament 39

Queen 40

Wireless Network 41

Robot versus Fan 43

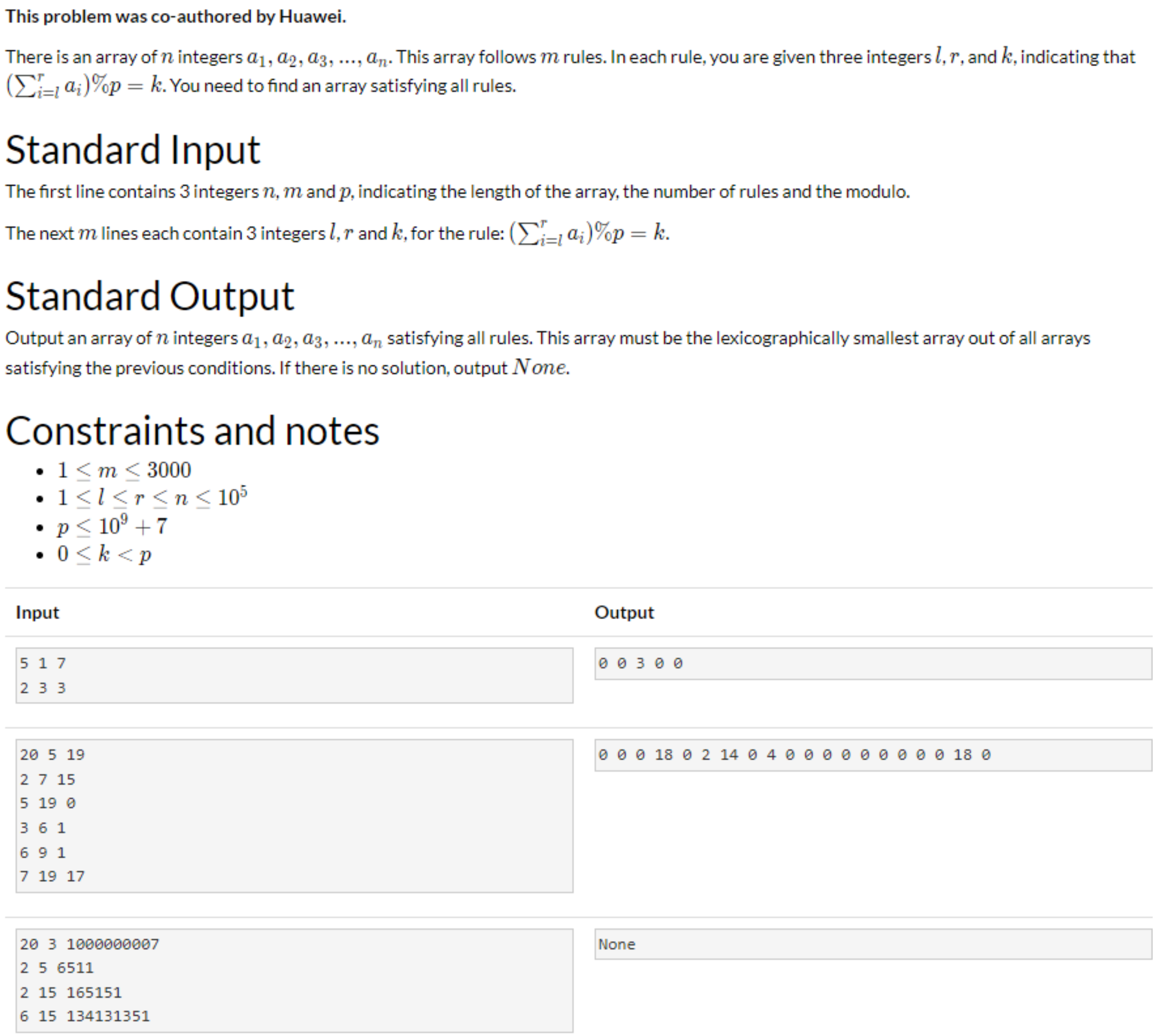
Tourist Taxi Service 46

Weird Reward 48

Random Mazes 50

Queries on a Tree 63

Array



This problem was co-authored by Huawei.

There is an array of n integers $a_1, a_2, a_3, \ldots, a_n$. This array follows m rules. In each rule, you are given three integers l, r, and k, indicating that $(\sum_{i=l}^{r} a_i)\%p = k$. You need to find an array satisfying all rules.

Standard Input

The first line contains 3 integers n, m and p, indicating the length of the array, the number of rules and the modulo.

The next m lines each contain 3 integers l, r and k, for the rule: $(\sum_{i=l}^{r} a_i)\%p = k$.

Standard Output

Output an array of n integers $a_1, a_2, a_3, \ldots, a_n$ satisfying all rules. This array must be the lexicographically smallest array out of all arrays satisfying the previous conditions. If there is no solution, output $None$.

Constraints and notes

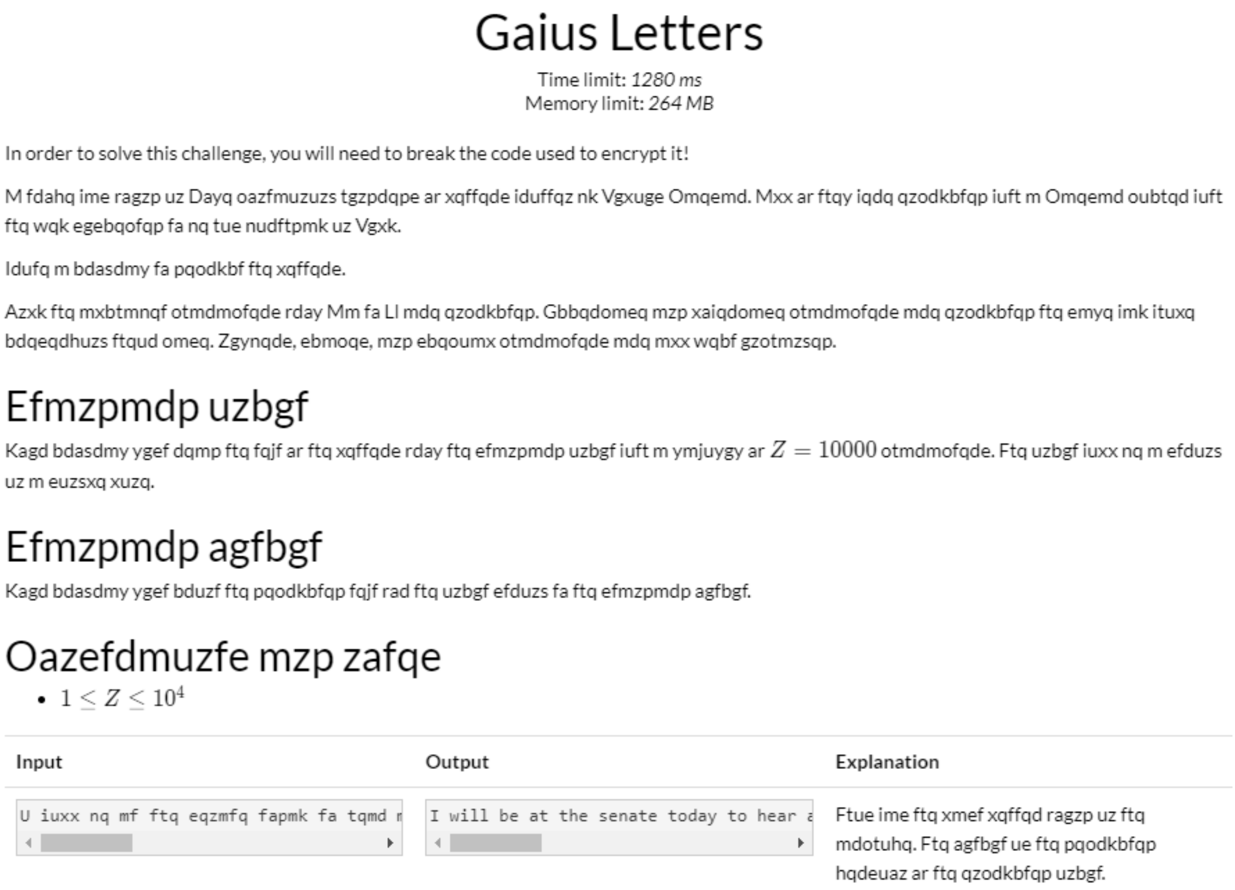
- $1 \leq m \leq 3000$
- $1 \leq l \leq r \leq n \leq 10^5$
- $p \leq 10^9 + 7$
- $0 \leq k < p$

Input	Output
5 1 7 2 3 3	0 0 3 0 0

20 5 19
2 7 15
5 19 0
3 6 1
6 9 1
7 19 17
	0 0 0 18 0 2 14 0 4 0 0 0 0 0 0 0 0 0 18 0

20 3 1000000007
2 5 6511
2 15 165151
6 15 134131351
	None

Gaius Letters



Gaius Letters

Time limit: *1280 ms*
Memory limit: *264 MB*

In order to solve this challenge, you will need to break the code used to encrypt it!

M fdahq ime ragzp uz Dayq oazfmuzuzs tgzpdqpe ar xqffqde iduffqz nk Vgxuge Omqemd. Mxx ar ftqy iqdq qzodkbfqp iuft m Omqemd oubtqd iuft ftq wqk egebqofqp fa nq tue nudftpmk uz Vgxk.

Idufq m bdasdmy fa pqodkbf ftq xqffqde.

Azxk ftq mxbtmnqf otmdmofqde rday Mm fa Ll mdq qzodkbfqp. Gbbqdomeq mzp xaiqdomeq otmdmofqde mdq qzodkbfqp ftq emyq imk ituxq bdqeqdhuzs ftqud omeq. Zgynqde, ebmoqe, mzp ebqoumx otmdmofqde mdq mxx wqbf gzotmzsqp.

Efmzpmdp uzbgf

Kagd bdasdmy ygef dqmp ftq fqjf ar ftq xqffqde rday ftq efmzpmdp uzbgf iuft m ymjuygy ar $Z = 10000$ otmdmofqde. Ftq uzbgf iuxx nq m efduzs uz m euzsxq xuzq.

Efmzpmdp agfbgf

Kagd bdasdmy ygef bduzf ftq pqodkbfqp fqjf rad ftq uzbgf efduzs fa ftq efmzpmdp agfbgf.

Oazefdmuzfe mzp zafqe

- $1 \leq Z \leq 10^4$

Input	Output	Explanation
`U iuxx nq mf ftq eqzmfq fapmk fa tqmd m`	`I will be at the senate today to hear a`	Ftue ime ftq xmef xqffqd ragzp uz ftq mdotuhq. Ftq agfbgf ue ftq pqodkbfqp hqdeuaz ar ftq qzodkbfqp uzbgf.

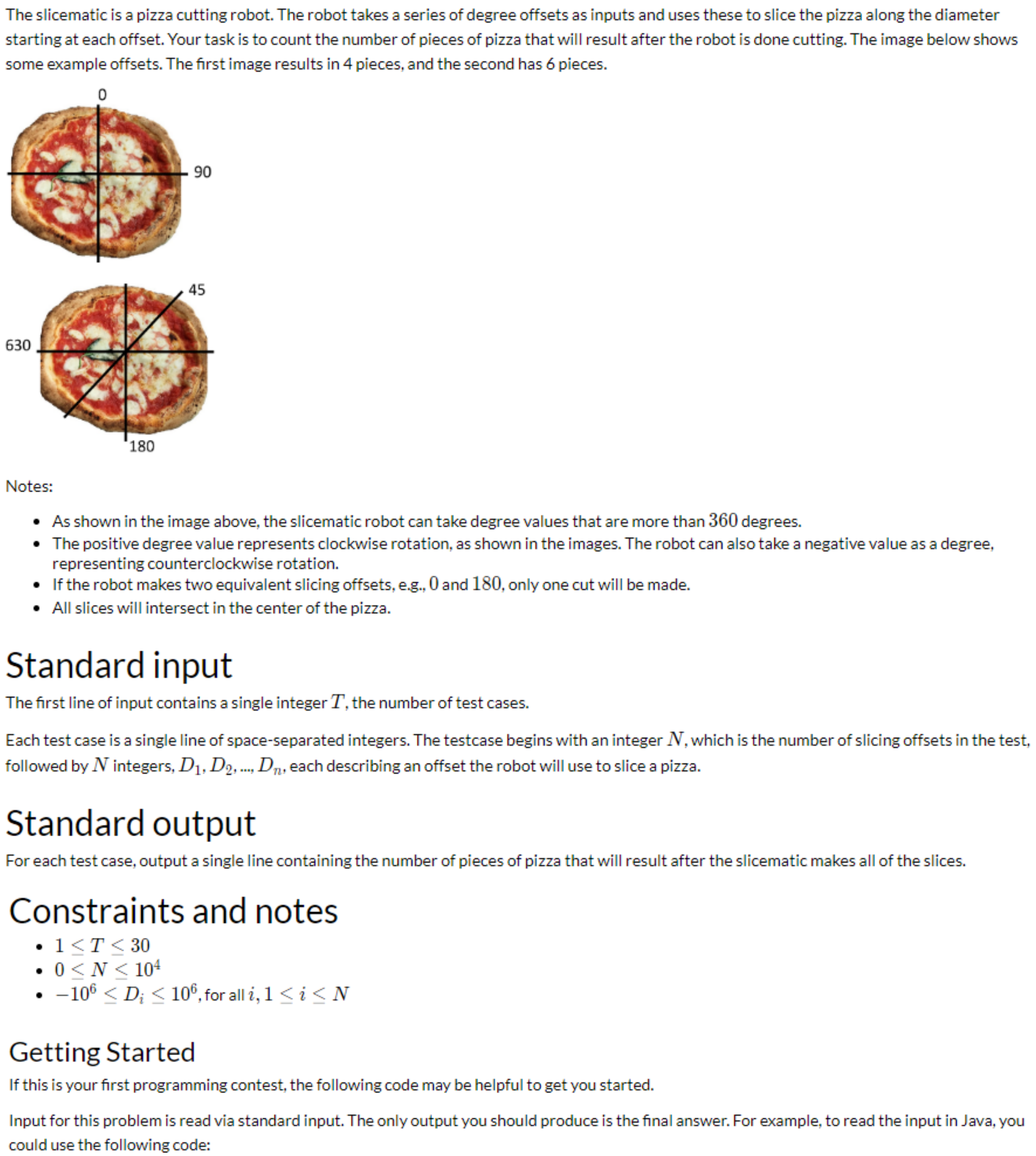
```
U iuxx nq mf ftq eqzmfq fapmk fa
tqmd m bqfufuaz rday Fuxxuge.
Omeeuge mzp Ndgfge tmhq nqqz
mofuzs efdmzsq. Etagxp nq nmow uz
fuyq rad puzzqd.
```

```
I will be at the senate today to
hear a petition from Tillius.
Cassius and Brutus have been
```

```
acting strange. Should be back in
time for dinner.
```

Ftue ime ftq xmef xqffqd ragzp uz ftq mdotuhq. Ftq agfbgf ue ftq pqodkbfqp hqdeuaz ar ftq qzodkbfqp uzbgf.

Pizza Cutter



The slicematic is a pizza cutting robot. The robot takes a series of degree offsets as inputs and uses these to slice the pizza along the diameter starting at each offset. Your task is to count the number of pieces of pizza that will result after the robot is done cutting. The image below shows some example offsets. The first image results in 4 pieces, and the second has 6 pieces.

Notes:

- As shown in the image above, the slicematic robot can take degree values that are more than 360 degrees.
- The positive degree value represents clockwise rotation, as shown in the images. The robot can also take a negative value as a degree, representing counterclockwise rotation.
- If the robot makes two equivalent slicing offsets, e.g., 0 and 180, only one cut will be made.
- All slices will intersect in the center of the pizza.

Standard input

The first line of input contains a single integer T, the number of test cases.

Each test case is a single line of space-separated integers. The testcase begins with an integer N, which is the number of slicing offsets in the test, followed by N integers, $D_1, D_2, ..., D_n$, each describing an offset the robot will use to slice a pizza.

Standard output

For each test case, output a single line containing the number of pieces of pizza that will result after the slicematic makes all of the slices.

Constraints and notes

- $1 \leq T \leq 30$
- $0 \leq N \leq 10^4$
- $-10^6 \leq D_i \leq 10^6$, for all $i, 1 \leq i \leq N$

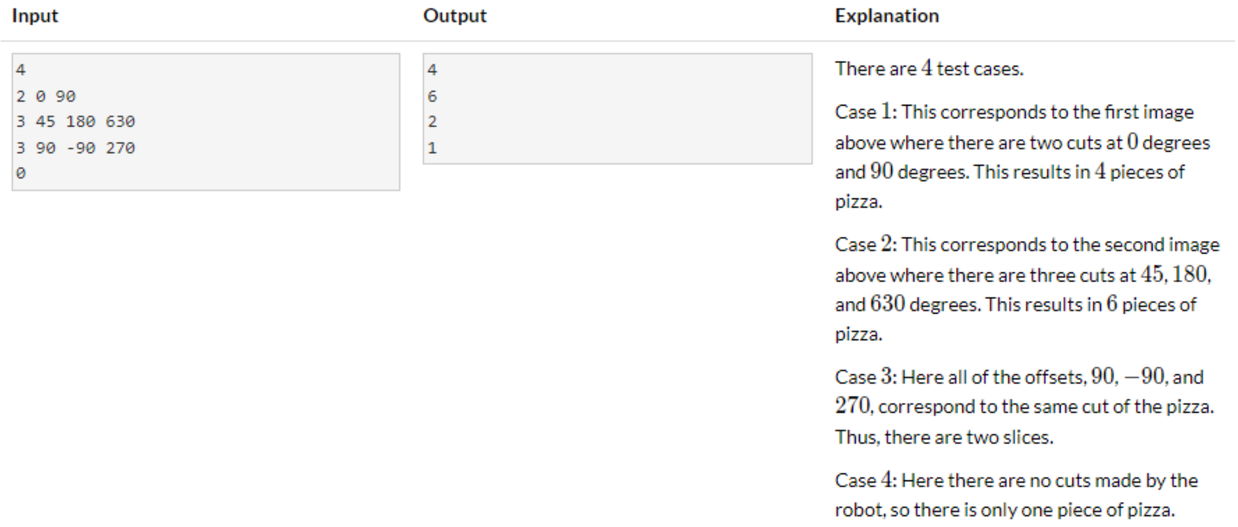
Getting Started

If this is your first programming contest, the following code may be helpful to get you started.

Input for this problem is read via standard input. The only output you should produce is the final answer. For example, to read the input in Java, you could use the following code:

```
// Don't place your source in a package
import java.util.*;
import java.lang.*;
import java.io.*;
```

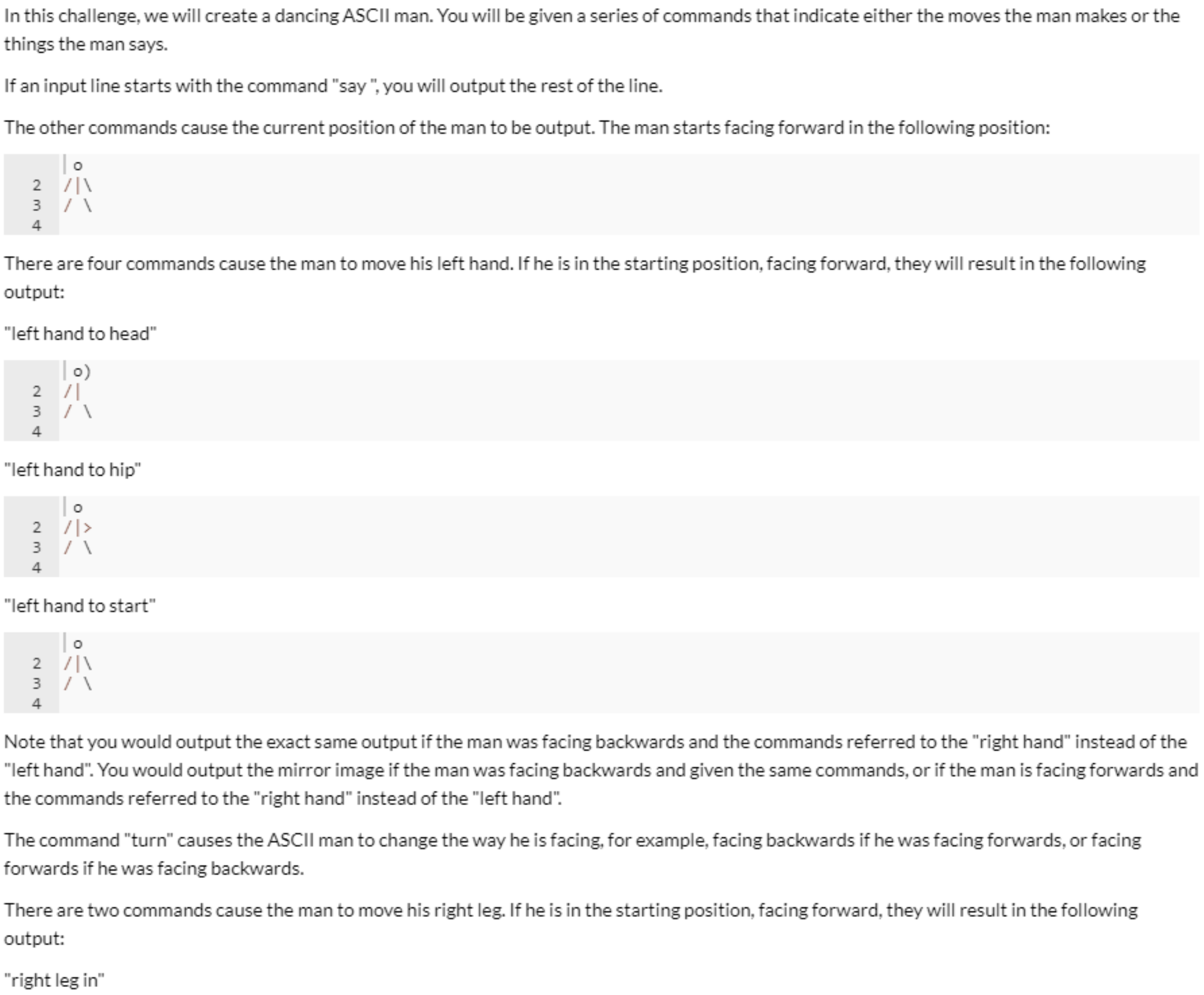
```
// Please name your class Main
class Main {
        public static void main (String[] args) throws java.lang.Exception {
    Scanner sc = new Scanner(System.in);

    // Read the number of testcases
    int T = sc.nextInt();
    // Process each test case
    for (int t = 0; t < T; t++) {
      // Read the number of offsets
      int N = sc.nextInt();

      int answer = 0;
      // Read each offset
      // TODO: You will need to figure out how to
      // process the offset and change the variable answer
      for (int n = 0; n < N; n++) {
        int D = sc.nextInt();
      }

      // Output your answer:
      System.out.println(answer);
    }

      }
}
```



Input	Output	Explanation

4
2 0 90
3 45 180 630
3 90 -90 270
0
	4
	6
	2
	1
		There are 4 test cases.
		Case 1: This corresponds to the first image above where there are two cuts at 0 degrees and 90 degrees. This results in 4 pieces of pizza.
		Case 2: This corresponds to the second image above where there are three cuts at 45, 180, and 630 degrees. This results in 6 pieces of pizza.
		Case 3: Here all of the offsets, 90, -90, and 270, correspond to the same cut of the pizza. Thus, there are two slices.
		Case 4: Here there are no cuts made by the robot, so there is only one piece of pizza.

ASCII Dancer



In this challenge, we will create a dancing ASCII man. You will be given a series of commands that indicate either the moves the man makes or the things the man says.

If an input line starts with the command "say ", you will output the rest of the line.

The other commands cause the current position of the man to be output. The man starts facing forward in the following position:

```
 o
/|\
/ \

```

There are four commands cause the man to move his left hand. If he is in the starting position, facing forward, they will result in the following output:

"left hand to head"

```
 o)
/|
/ \

```

"left hand to hip"

```
 o
/|>
/ \

```

"left hand to start"

```
 o
/|\
/ \

```

Note that you would output the exact same output if the man was facing backwards and the commands referred to the "right hand" instead of the "left hand". You would output the mirror image if the man was facing backwards and given the same commands, or if the man is facing forwards and the commands referred to the "right hand" instead of the "left hand".

The command "turn" causes the ASCII man to change the way he is facing, for example, facing backwards if he was facing forwards, or facing forwards if he was facing backwards.

There are two commands cause the man to move his right leg. If he is in the starting position, facing forward, they will result in the following output:

"right leg in"

```
 o
/|\
< \
```

"right leg out"

```
 o
/|\
/ \
```

If a previous limb has moved, it stays in the same position until it moves again. That allows for sequences of moves like the one below. Here we are starting from the starting position, facing forwards:

"right hand to hip"

```
 o
<|\
/ \
```

"left hand to head"

```
 o)
<|
/ \
```

"left leg in"

```
 o)
<|
/ >
```

Standard input

The input begins with an integer T giving the number of test cases.

Each test case begins with an integer d that gives the number of commands in the dance.

The next d lines give a command in from the following list:

- "left hand to head"
- "left hand to hip"
- "left hand to start"
- "left leg in"
- "left leg out"
- "right hand to head"
- "right hand to hip"
- "right hand to start"
- "right leg in"
- "right leg out"
- "turn"
- "say [words]", where [words] is a string of made up of letters, numbers, punctuation, and spaces.

Standard output

Output consists of either three lines, with three characters each, in response to a movement command, or one line in response to a "say" command.

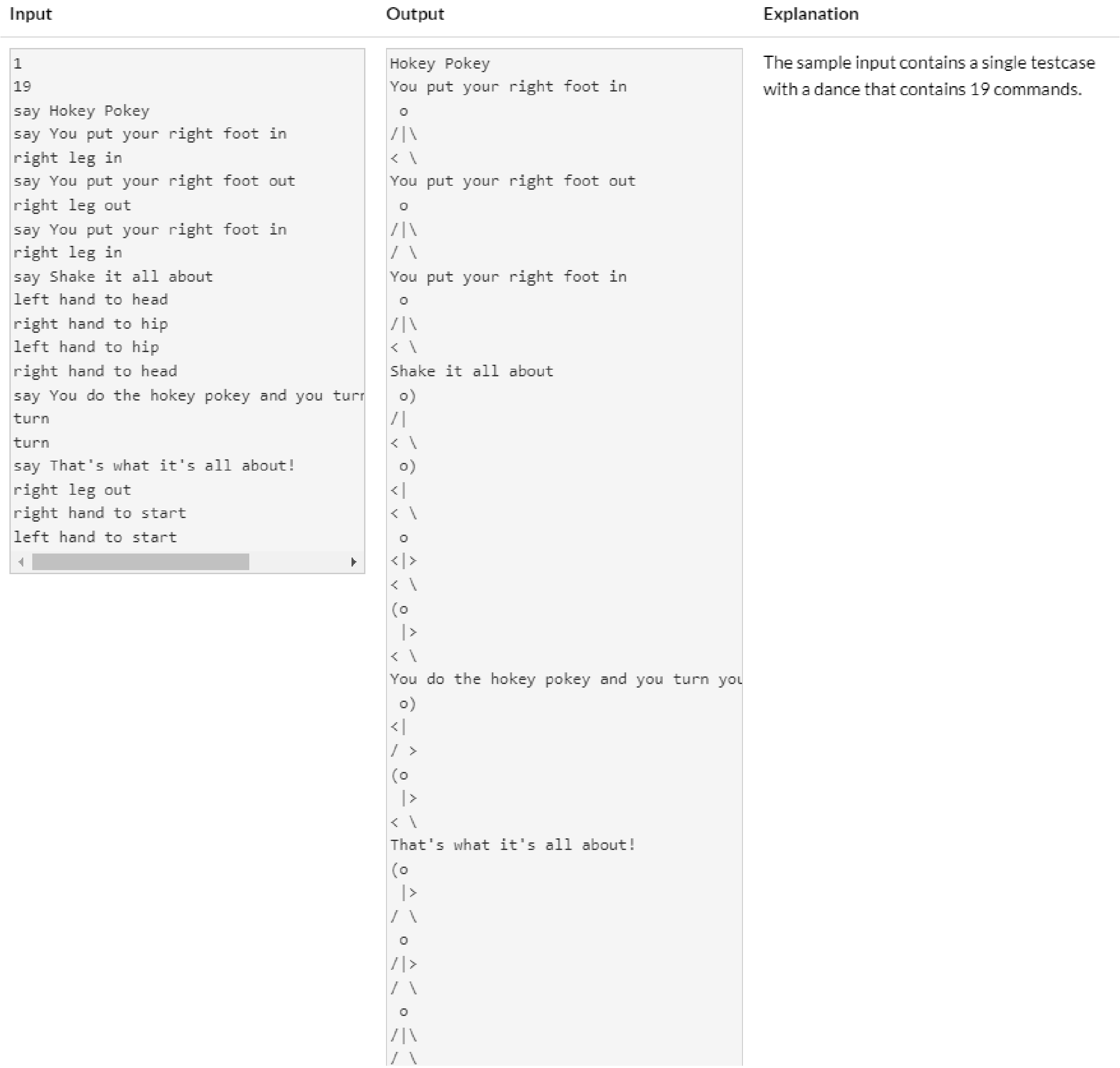
Constraints and notes

- $1 \leq T \leq 10$
- $3 \leq d \leq 400$

The number of characters to be output in any "say" command will be between 1 and 80, inclusive.

It is possible that a move command will not change the position of the ASCII Man, for example if the same command is given twice in a row. Even if the position does not change as a result of a movement command, you still should produce the output showing the current position.

The input will never ask to have both the left leg in and the right leg in.



Input

```
1
19
say Hokey Pokey
say You put your right foot in
right leg in
say You put your right foot out
right leg out
say You put your right foot in
right leg in
say Shake it all about
left hand to head
right hand to hip
left hand to hip
right hand to head
say You do the hokey pokey and you turn
turn
turn
say That's what it's all about!
right leg out
right hand to start
left hand to start
```

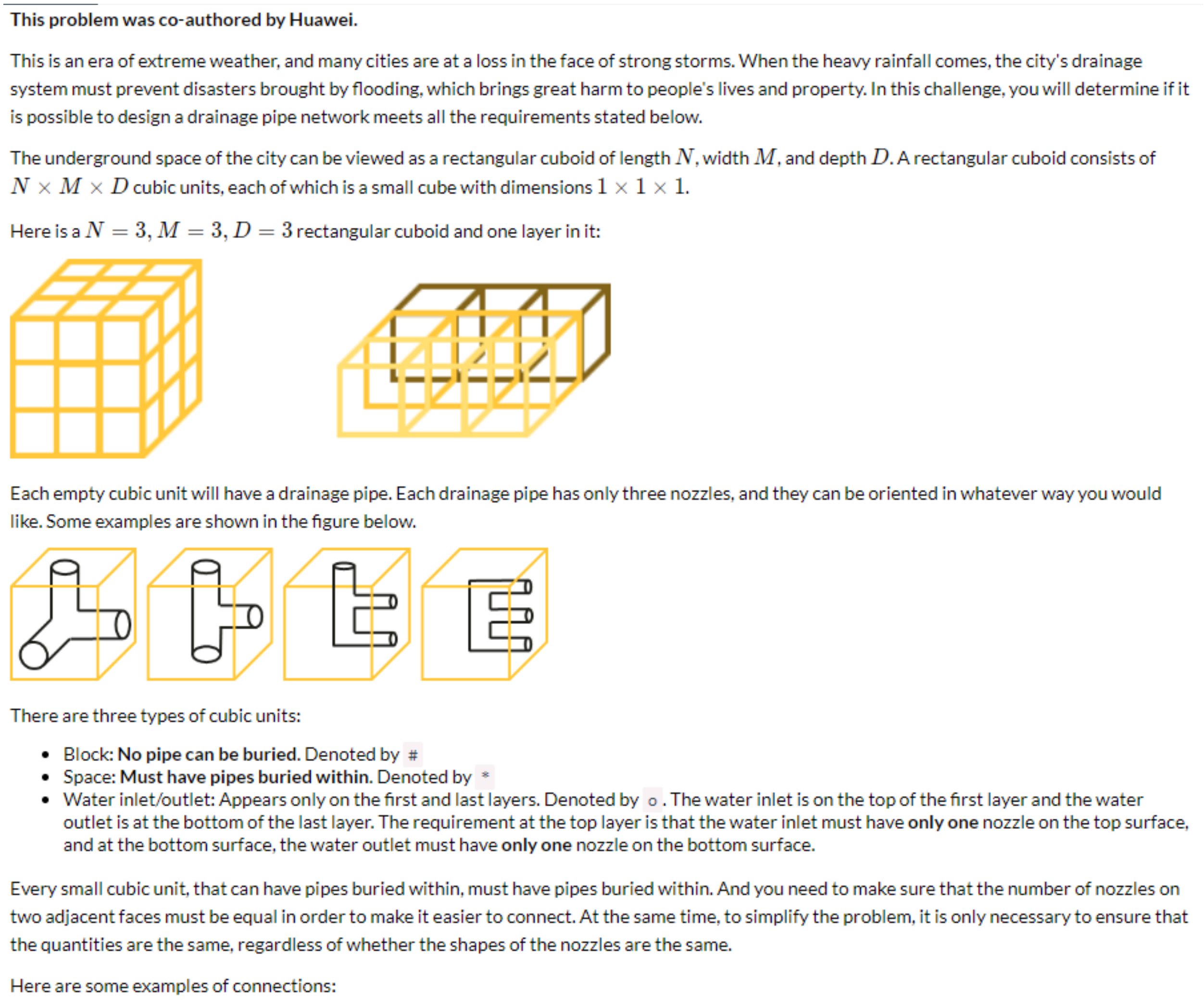
Output

```
Hokey Pokey
You put your right foot in
 o
/|\
< \
You put your right foot out
 o
/|\
/ \
You put your right foot in
 o
/|\
< \
Shake it all about
 o)
/|
< \
 o)
<|
< \
 o
<|>
< \
(o
 |>
< \
You do the hokey pokey and you turn you
 o)
<|
/ >
(o
 |>
< \
That's what it's all about!
(o
 |>
/ \
 o
/|>
/ \
 o
/|\
/ \
```

Explanation

The sample input contains a single testcase with a dance that contains 19 commands.

Pipeline



This problem was co-authored by Huawei.

This is an era of extreme weather, and many cities are at a loss in the face of strong storms. When the heavy rainfall comes, the city's drainage system must prevent disasters brought by flooding, which brings great harm to people's lives and property. In this challenge, you will determine if it is possible to design a drainage pipe network meets all the requirements stated below.

The underground space of the city can be viewed as a rectangular cuboid of length N, width M, and depth D. A rectangular cuboid consists of $N \times M \times D$ cubic units, each of which is a small cube with dimensions $1 \times 1 \times 1$.

Here is a $N = 3, M = 3, D = 3$ rectangular cuboid and one layer in it:

Each empty cubic unit will have a drainage pipe. Each drainage pipe has only three nozzles, and they can be oriented in whatever way you would like. Some examples are shown in the figure below.

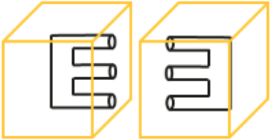
There are three types of cubic units:

- Block: **No pipe can be buried.** Denoted by `#`
- Space: **Must have pipes buried within.** Denoted by `*`
- Water inlet/outlet: Appears only on the first and last layers. Denoted by `o`. The water inlet is on the top of the first layer and the water outlet is at the bottom of the last layer. The requirement at the top layer is that the water inlet must have **only one** nozzle on the top surface, and at the bottom surface, the water outlet must have **only one** nozzle on the bottom surface.

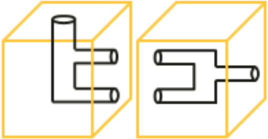
Every small cubic unit, that can have pipes buried within, must have pipes buried within. And you need to make sure that the number of nozzles on two adjacent faces must be equal in order to make it easier to connect. At the same time, to simplify the problem, it is only necessary to ensure that the quantities are the same, regardless of whether the shapes of the nozzles are the same.

Here are some examples of connections:

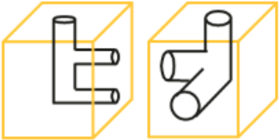
Legal:



Legal:



Illegal:



Standard input

The first line contains one integer T – the number of test cases in input.

The first line of each case contains three integers N, M, D – length, width, and depth of the rectangular cuboid.

Next there are $N \times M$ matrices for each of the D levels. The level appears on N lines with M characters each. Each character of a matrix is either `*`, `#` or `o`. After each level of the matrix, there is a blank line.

Standard Output

Output T lines, each of which contains an evaluation of the corresponding test case. For each case, if it is possible to create a design that meets all the requirements, output `YES`, else output `NO`.

Constraints and notes

- $1 \le T \le 15$
- $1 \le N, M \le 50$
- $2 \le D \le 50$
- $\sum N \times M \times D \le 100,000$

Notes:

1. No nozzles are allowed on surfaces that are not adjacent to any other small cubic unit except for the inlet and outlet.
2. The water inlet and the water outlet do not necessarily need to be connected.
3. A closed area consisting of multiple pipes is permitted (as shown in the first legal connection example).

Input

```
1
3 3 3
**o
***
***

#**
**#
***

**o
*o*
***
```

Output

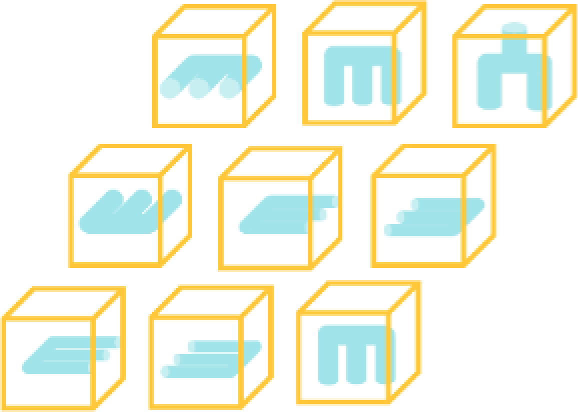
```
YES
```

One of the feasible drainage designs is shown below.

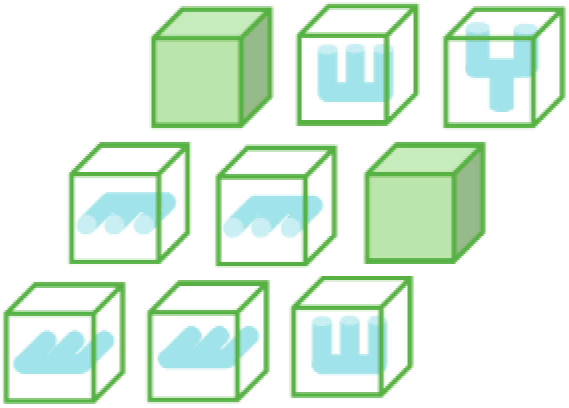
The first layer:

One of the feasible drainage designs is shown below.

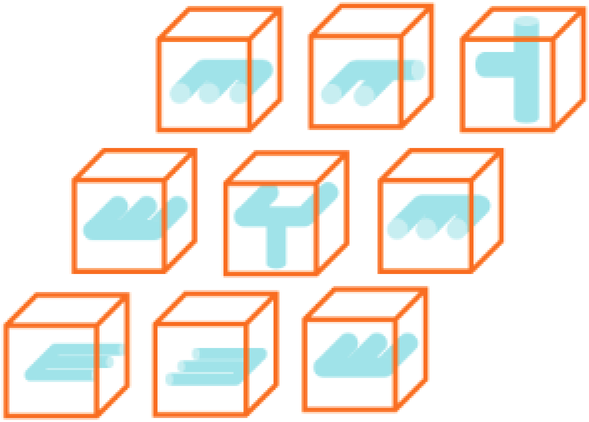
The first layer:



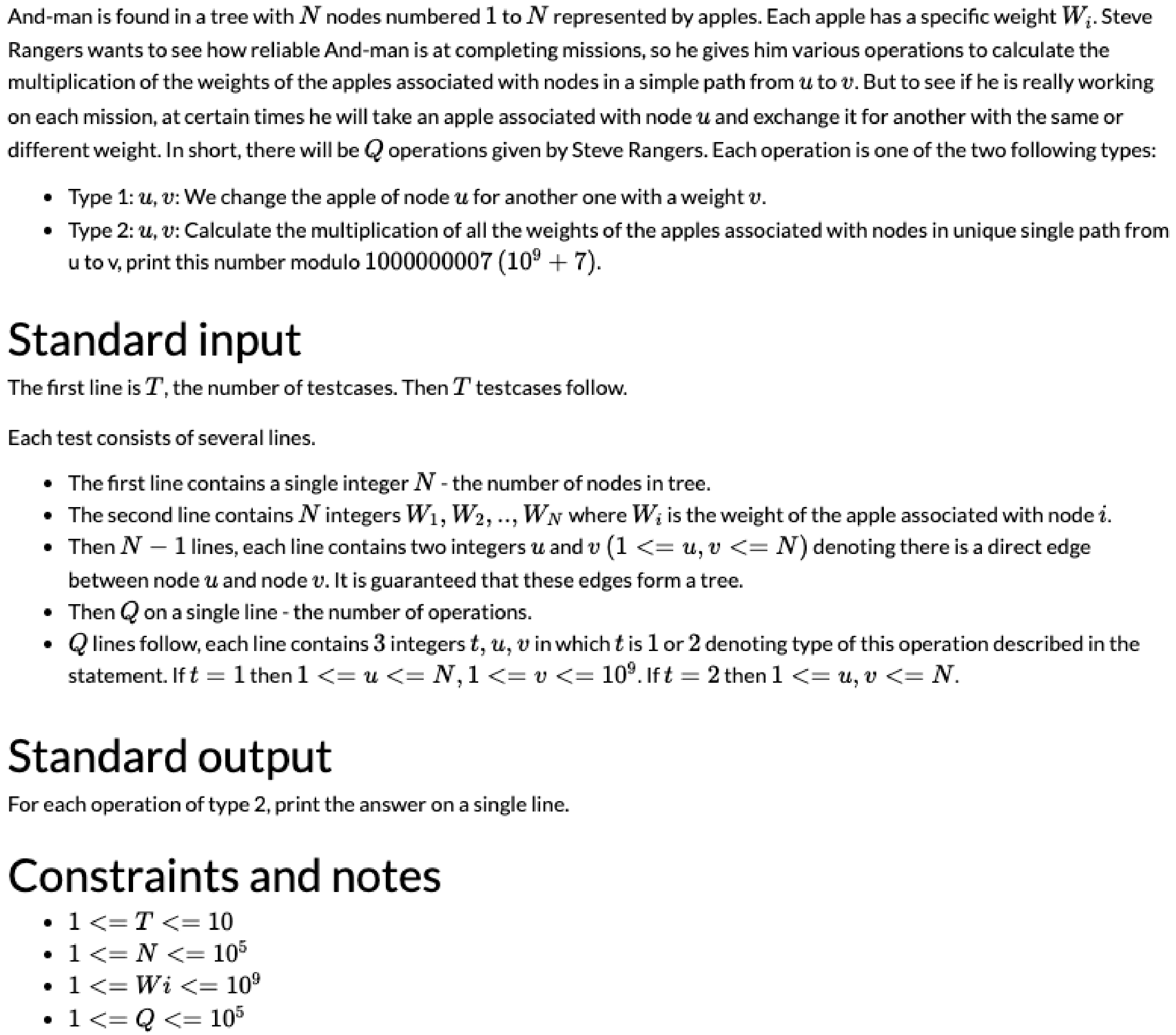
The second layer:



The third layer:



ANDman



And-man is found in a tree with N nodes numbered 1 to N represented by apples. Each apple has a specific weight W_i. Steve Rangers wants to see how reliable And-man is at completing missions, so he gives him various operations to calculate the multiplication of the weights of the apples associated with nodes in a simple path from u to v. But to see if he is really working on each mission, at certain times he will take an apple associated with node u and exchange it for another with the same or different weight. In short, there will be Q operations given by Steve Rangers. Each operation is one of the two following types:

- Type 1: u, v: We change the apple of node u for another one with a weight v.
- Type 2: u, v: Calculate the multiplication of all the weights of the apples associated with nodes in unique single path from u to v, print this number modulo 1000000007 $(10^9 + 7)$.

Standard input

The first line is T, the number of testcases. Then T testcases follow.

Each test consists of several lines.

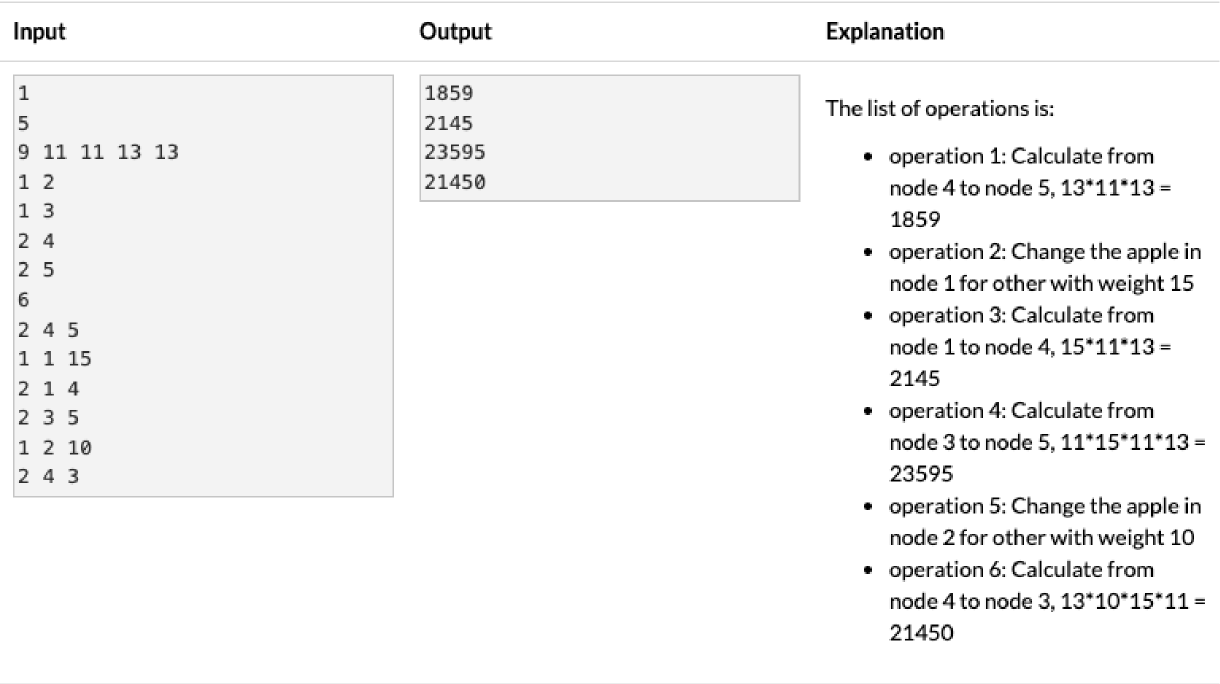
- The first line contains a single integer N - the number of nodes in tree.
- The second line contains N integers $W_1, W_2, .., W_N$ where W_i is the weight of the apple associated with node i.
- Then $N - 1$ lines, each line contains two integers u and v $(1 <= u, v <= N)$ denoting there is a direct edge between node u and node v. It is guaranteed that these edges form a tree.
- Then Q on a single line - the number of operations.
- Q lines follow, each line contains 3 integers t, u, v in which t is 1 or 2 denoting type of this operation described in the statement. If $t = 1$ then $1 <= u <= N, 1 <= v <= 10^9$. If $t = 2$ then $1 <= u, v <= N$.

Standard output

For each operation of type 2, print the answer on a single line.

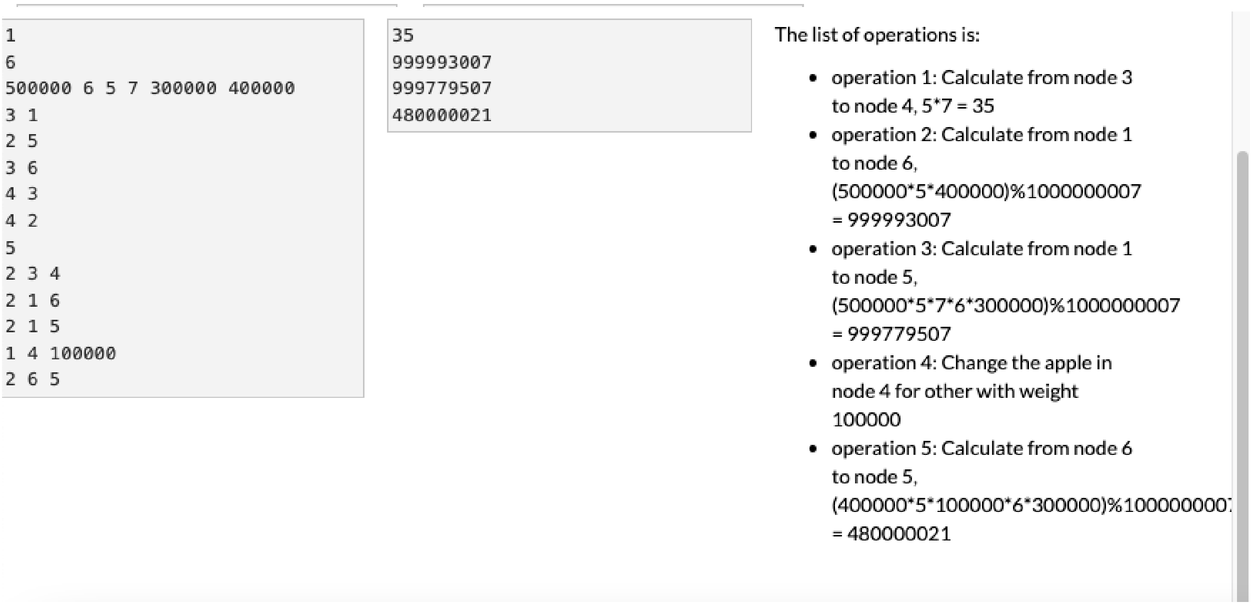
Constraints and notes

- $1 <= T <= 10$
- $1 <= N <= 10^5$
- $1 <= Wi <= 10^9$
- $1 <= Q <= 10^5$



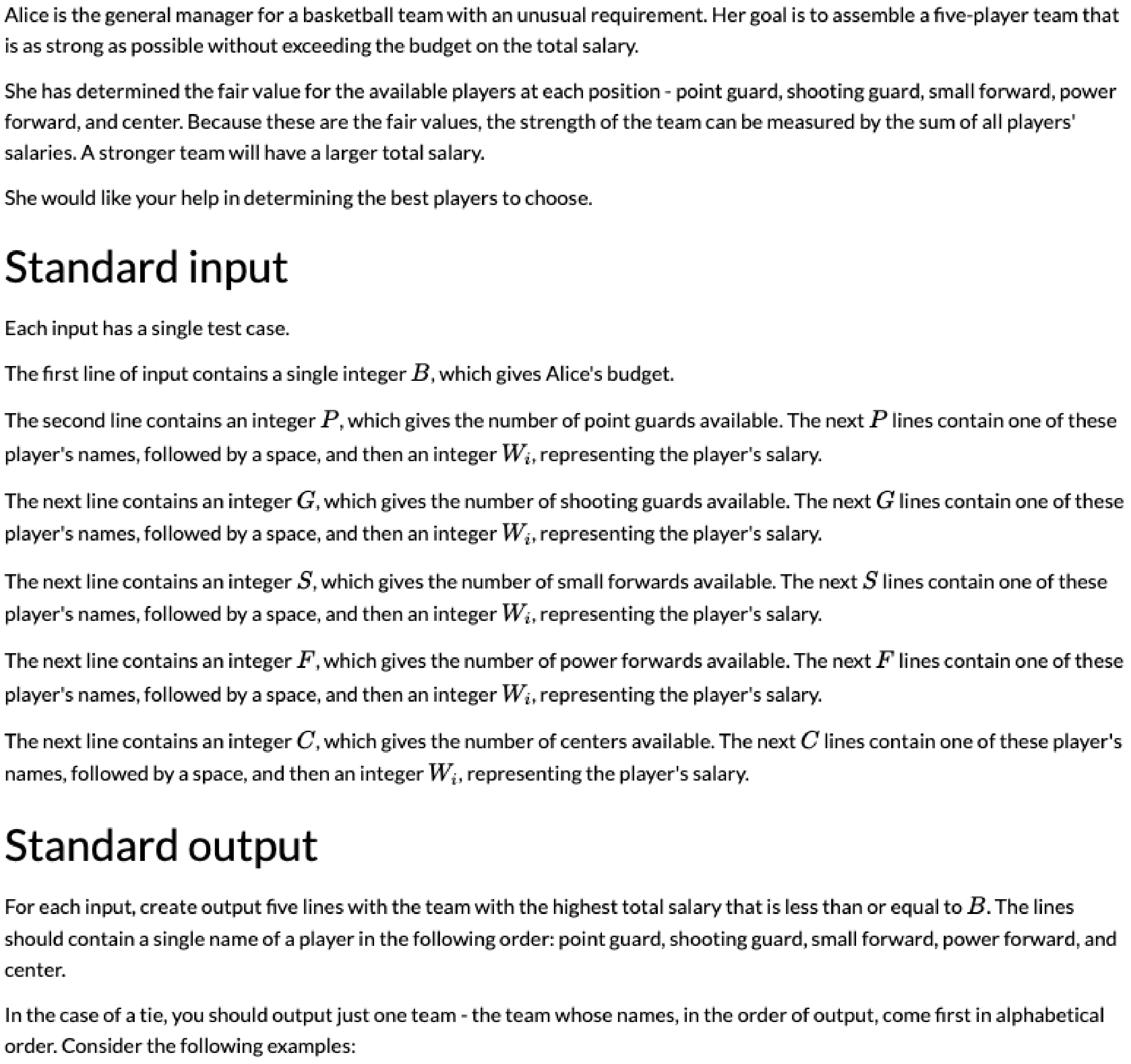
Input	Output	Explanation

1
5
9 11 11 13 13
1 2
1 3
2 4
2 5
6
2 4 5
1 1 15
2 1 4
2 3 5
1 2 10
2 4 3
	1859
	2145
	23595
	21450
		The list of operations is:
		- operation 1: Calculate from node 4 to node 5, 13*11*13 = 1859
		- operation 2: Change the apple in node 1 for other with weight 15
		- operation 3: Calculate from node 1 to node 4, 15*11*13 = 2145
		- operation 4: Calculate from node 3 to node 5, 11*15*11*13 = 23595
		- operation 5: Change the apple in node 2 for other with weight 10
		- operation 6: Calculate from node 4 to node 3, 13*10*15*11 = 21450



1
6
500000 6 5 7 300000 400000
3 1
2 5
3 6
4 3
4 2
5
2 3 4
2 1 6
2 1 5
1 4 100000
2 6 5
	35
	999993007
	999779507
	480000021
		The list of operations is:
		- operation 1: Calculate from node 3 to node 4, 5*7 = 35
		- operation 2: Calculate from node 1 to node 6, (500000*5*400000)%1000000007 = 999993007
		- operation 3: Calculate from node 1 to node 5, (500000*5*7*6*300000)%1000000007 = 999779507
		- operation 4: Change the apple in node 4 for other with weight 100000
		- operation 5: Calculate from node 6 to node 5, (400000*5*100000*6*300000)%1000000007 = 480000021

Dream Team



Alice is the general manager for a basketball team with an unusual requirement. Her goal is to assemble a five-player team that is as strong as possible without exceeding the budget on the total salary.

She has determined the fair value for the available players at each position - point guard, shooting guard, small forward, power forward, and center. Because these are the fair values, the strength of the team can be measured by the sum of all players' salaries. A stronger team will have a larger total salary.

She would like your help in determining the best players to choose.

Standard input

Each input has a single test case.

The first line of input contains a single integer B, which gives Alice's budget.

The second line contains an integer P, which gives the number of point guards available. The next P lines contain one of these player's names, followed by a space, and then an integer W_i, representing the player's salary.

The next line contains an integer G, which gives the number of shooting guards available. The next G lines contain one of these player's names, followed by a space, and then an integer W_i, representing the player's salary.

The next line contains an integer S, which gives the number of small forwards available. The next S lines contain one of these player's names, followed by a space, and then an integer W_i, representing the player's salary.

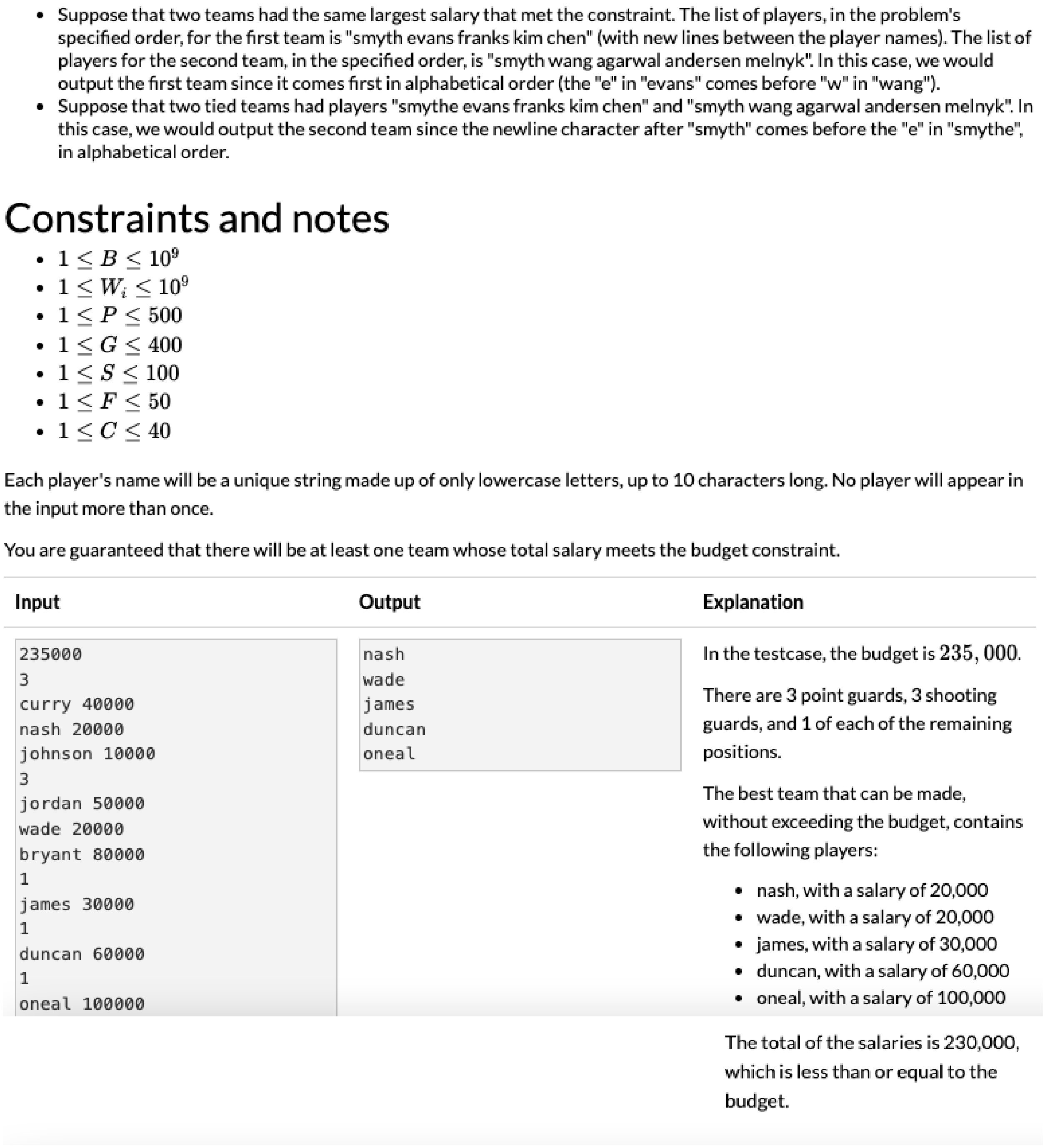
The next line contains an integer F, which gives the number of power forwards available. The next F lines contain one of these player's names, followed by a space, and then an integer W_i, representing the player's salary.

The next line contains an integer C, which gives the number of centers available. The next C lines contain one of these player's names, followed by a space, and then an integer W_i, representing the player's salary.

Standard output

For each input, create output five lines with the team with the highest total salary that is less than or equal to B. The lines should contain a single name of a player in the following order: point guard, shooting guard, small forward, power forward, and center.

In the case of a tie, you should output just one team - the team whose names, in the order of output, come first in alphabetical order. Consider the following examples:



- Suppose that two teams had the same largest salary that met the constraint. The list of players, in the problem's specified order, for the first team is "smyth evans franks kim chen" (with new lines between the player names). The list of players for the second team, in the specified order, is "smyth wang agarwal andersen melnyk". In this case, we would output the first team since it comes first in alphabetical order (the "e" in "evans" comes before "w" in "wang").
- Suppose that two tied teams had players "smythe evans franks kim chen" and "smyth wang agarwal andersen melnyk". In this case, we would output the second team since the newline character after "smyth" comes before the "e" in "smythe", in alphabetical order.

Constraints and notes

- $1 \leq B \leq 10^9$
- $1 \leq W_i \leq 10^9$
- $1 \leq P \leq 500$
- $1 \leq G \leq 400$
- $1 \leq S \leq 100$
- $1 \leq F \leq 50$
- $1 \leq C \leq 40$

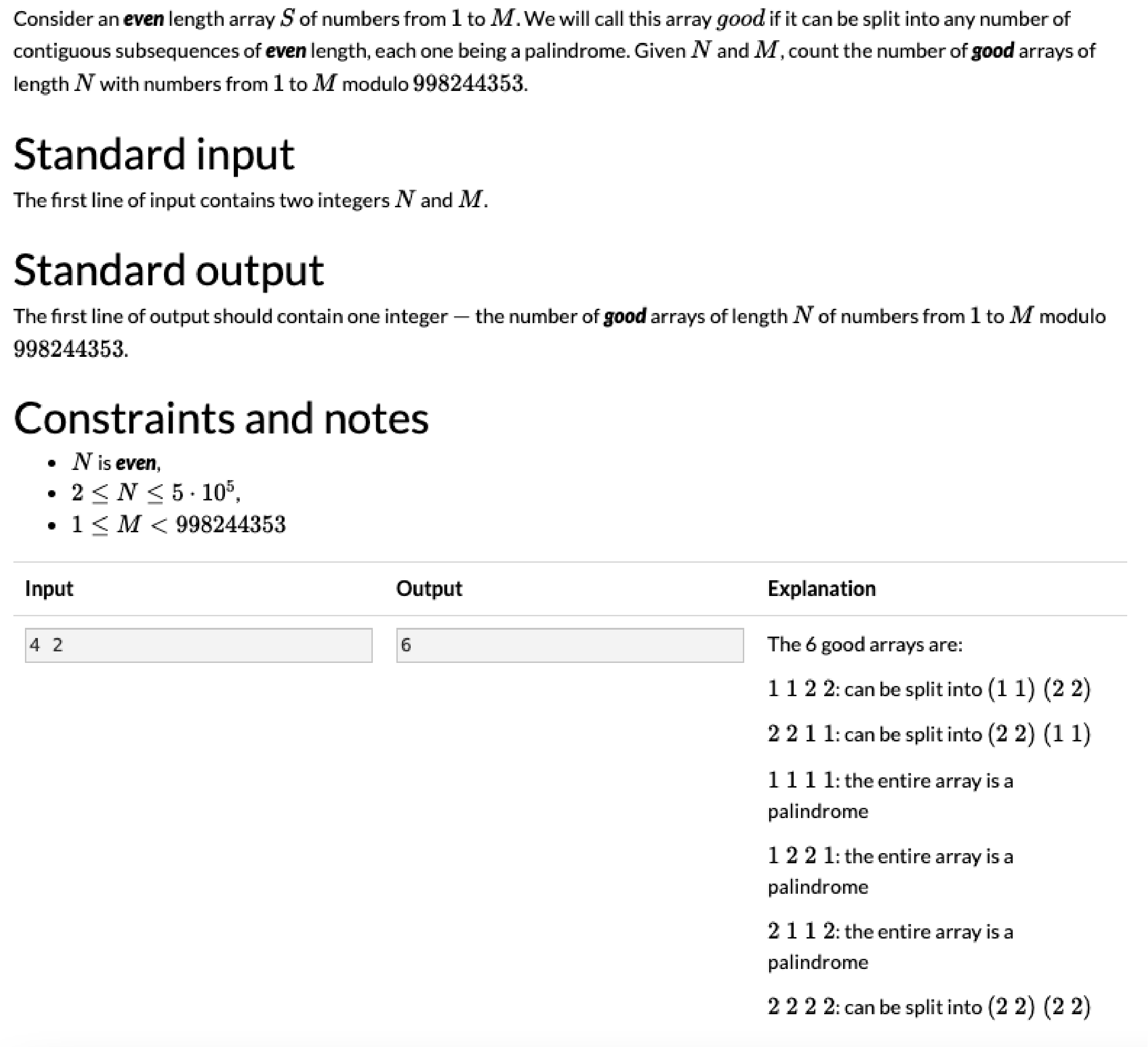
Each player's name will be a unique string made up of only lowercase letters, up to 10 characters long. No player will appear in the input more than once.

You are guaranteed that there will be at least one team whose total salary meets the budget constraint.

Input	Output	Explanation

`235000`
`3`
`curry 40000`
`nash 20000`
`johnson 10000`
`3`
`jordan 50000`
`wade 20000`
`bryant 80000`
`1`
`james 30000`
`1`
`duncan 60000`
`1`
`oneal 100000`
	`nash`
	`wade`
	`james`
	`duncan`
	`oneal`
		In the testcase, the budget is $235,000$.
		There are 3 point guards, 3 shooting guards, and 1 of each of the remaining positions.
		The best team that can be made, without exceeding the budget, contains the following players:
		• nash, with a salary of 20,000
		• wade, with a salary of 20,000
		• james, with a salary of 30,000
		• duncan, with a salary of 60,000
		• oneal, with a salary of 100,000
		The total of the salaries is 230,000, which is less than or equal to the budget.

Palindrome Cutting



Consider an ***even*** length array S of numbers from 1 to M. We will call this array $good$ if it can be split into any number of contiguous subsequences of ***even*** length, each one being a palindrome. Given N and M, count the number of ***good*** arrays of length N with numbers from 1 to M modulo 998244353.

Standard input

The first line of input contains two integers N and M.

Standard output

The first line of output should contain one integer — the number of ***good*** arrays of length N of numbers from 1 to M modulo 998244353.

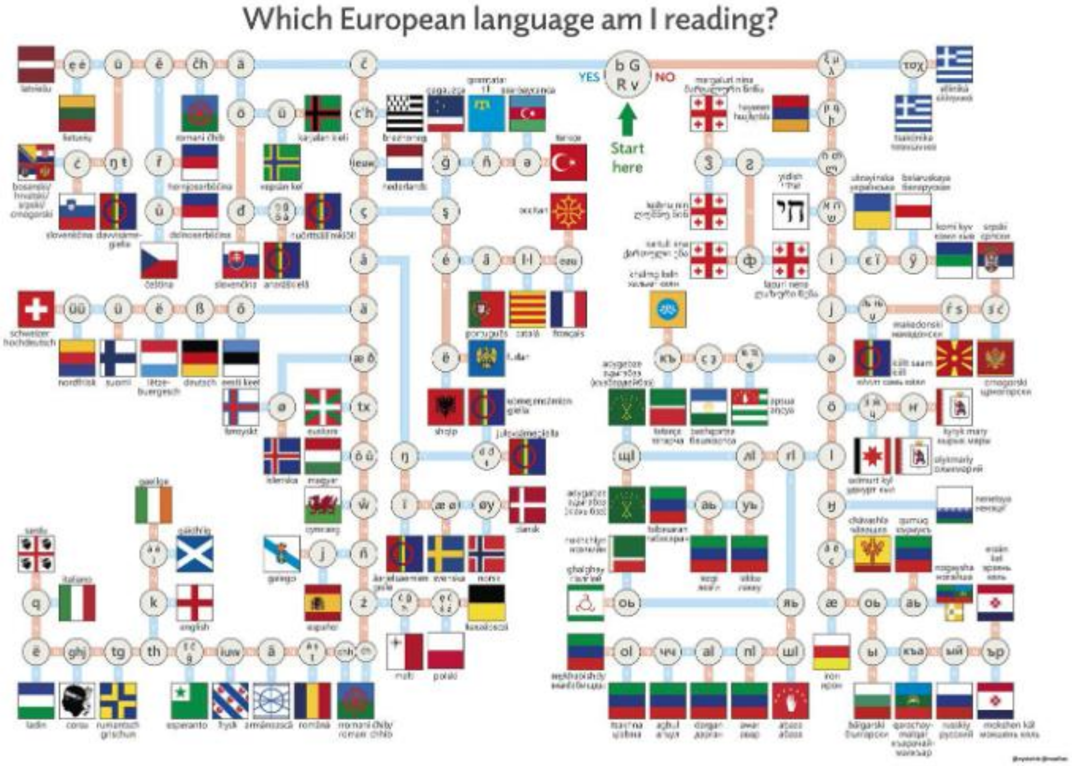
Constraints and notes

- N is ***even***,
- $2 \leq N \leq 5 \cdot 10^5$,
- $1 \leq M < 998244353$

Input	Output	Explanation
4 2	6	The 6 good arrays are: $1\ 1\ 2\ 2$: can be split into $(1\ 1)\ (2\ 2)$ $2\ 2\ 1\ 1$: can be split into $(2\ 2)\ (1\ 1)$ $1\ 1\ 1\ 1$: the entire array is a palindrome $1\ 2\ 2\ 1$: the entire array is a palindrome $2\ 1\ 1\ 2$: the entire array is a palindrome $2\ 2\ 2\ 2$: can be split into $(2\ 2)\ (2\ 2)$

What Language Am I Speaking?

The decision tree below can be used to determine the language used in a phrase based on the characters present in the text. If a character is present in the text, we should follow the "YES" branch of the tree. However, if a character is not present, it may be because it is simply not present in the phrase. Therefore, the language might be in either the "YES" or the "NO" branch of the tree.



Your task in this challenge is to determine the language that might be used based on a simpler type of decision tree.

Standard input

The first line of input contains two space-separated integers n, p, where n is the number of nodes in the decision tree, and p is the number of phrases that we must analyze.

The next n lines describe a node in a tree.

If the node is an internal node, it will have the following format: `I [id] [character] [YES_id] [NO_id]`, where `[id]` is an integer that uniquely identifies the node, `[character]` is a single character, `[YES_ID]` is the id of the child node on the "YES" branch, and `[NO_id]` is the id of the child node on the "NO" branch. The presence or absence of `[character]` in a phrase determines which branch or branches you should follow.

If the node is a leaf node, it will have the following format: `L [id] [language]`, where `[id]` is an integer that uniquely identifies the node, and `[language]` is the name of the language in the leaf.

The next p lines contain phrases that should be analyzed.

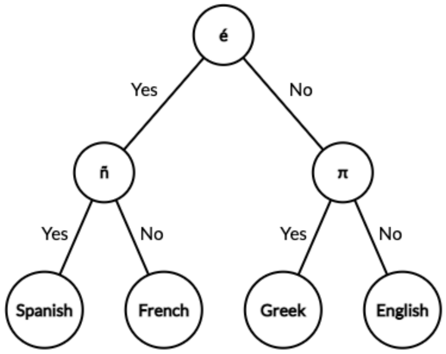
Standard output

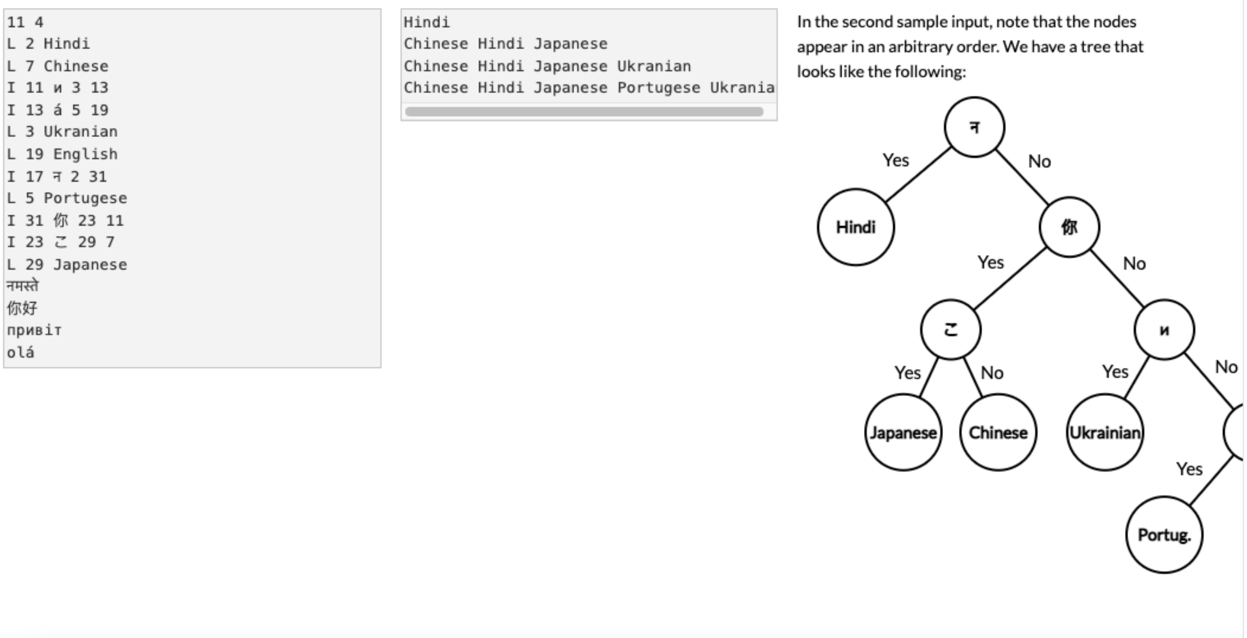
For each phrase, output a list of languages that the phrase could be using, according to the decision tree. The list should be output in alphabetical order.

Constraints and notes

- $0 \leq n \leq 10^4$
- $0 \leq p \leq 10^4$
- $1 \leq id \leq 10^7$, for all id's
- The name of each language will contain an initial capital and the rest of the characters will be lowercase letters. The length of a language name will be no more than 20 characters.
- Each phrase will be at most 1000 characters.
- The input files are encoded using UTF-8 encoding.
- If two characters have different UTF-8 encodings, they should be considered distinct. For example, "A" (UTF-8 character 0x41) is not the same as "a" (UTF-8 character 0x61). Similarly, "ĸ" (UTF-8 character 0x138) is not the same as "κ" (UTF-8 character 0x3BA).

Input	Output	Explanation

`7 2`
`I 1 é 2 3`
`I 2 ñ 4 5`
`I 3 π 6 7`
`L 4 Spanish`
`L 5 French`
`L 6 Greek`
`L 7 English`
`después de un año`
`un año`
	`Spanish`
	`English Greek Spanish`
		In this input, we have a tree that looks like the following:
		
		 é
		Yes
		No
		ñ
		π
		Yes
		No
		Yes
		No
		Spanish
		French
		Greek
		English
		The phrase "después de un año" must be in Spanish according to the tree, since neither Greek nor English use the é character, and French does not use the ñ character.
		The phrase "un año" could be in English, Greek, or Spanish, since we cannot be sure, based on the tree, that Greek and English do not use the ñ symbol.

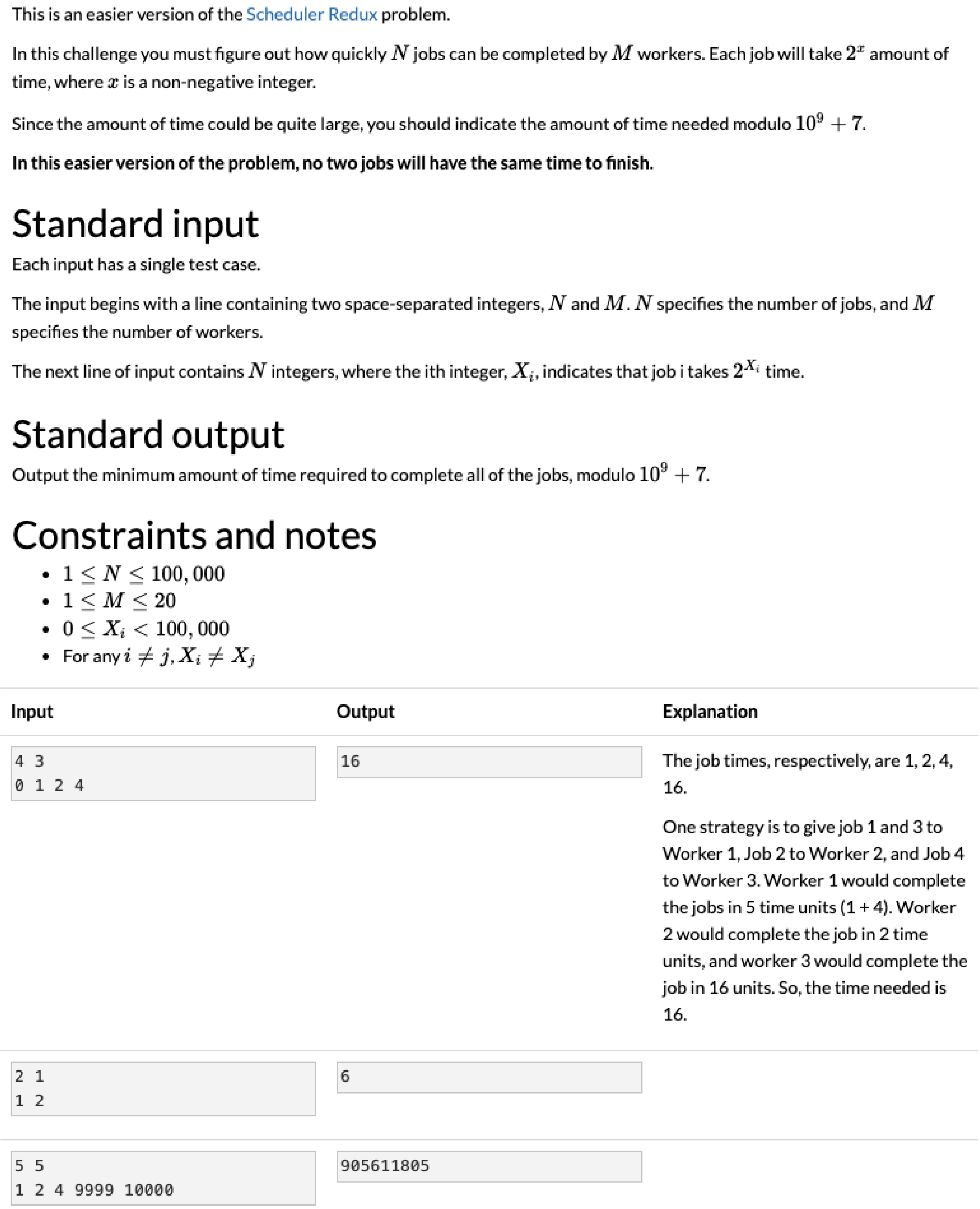


```
11 4
L 2 Hindi
L 7 Chinese
I 11 и 3 13
I 13 á 5 19
L 3 Ukranian
L 19 English
I 17 न 2 31
L 5 Portugese
I 31 你 23 11
I 23 こ 29 7
L 29 Japanese
नमस्ते
你好
привіт
olá
```

```
Hindi
Chinese Hindi Japanese
Chinese Hindi Japanese Ukranian
Chinese Hindi Japanese Portugese Ukrania
```

In the second sample input, note that the nodes appear in an arbitrary order. We have a tree that looks like the following:

Scheduler



This is an easier version of the Scheduler Redux problem.

In this challenge you must figure out how quickly N jobs can be completed by M workers. Each job will take 2^x amount of time, where x is a non-negative integer.

Since the amount of time could be quite large, you should indicate the amount of time needed modulo $10^9 + 7$.

In this easier version of the problem, no two jobs will have the same time to finish.

Standard input

Each input has a single test case.

The input begins with a line containing two space-separated integers, N and M. N specifies the number of jobs, and M specifies the number of workers.

The next line of input contains N integers, where the ith integer, X_i, indicates that job i takes 2^{X_i} time.

Standard output

Output the minimum amount of time required to complete all of the jobs, modulo $10^9 + 7$.

Constraints and notes

- $1 \leq N \leq 100,000$
- $1 \leq M \leq 20$
- $0 \leq X_i < 100,000$
- For any $i \neq j, X_i \neq X_j$

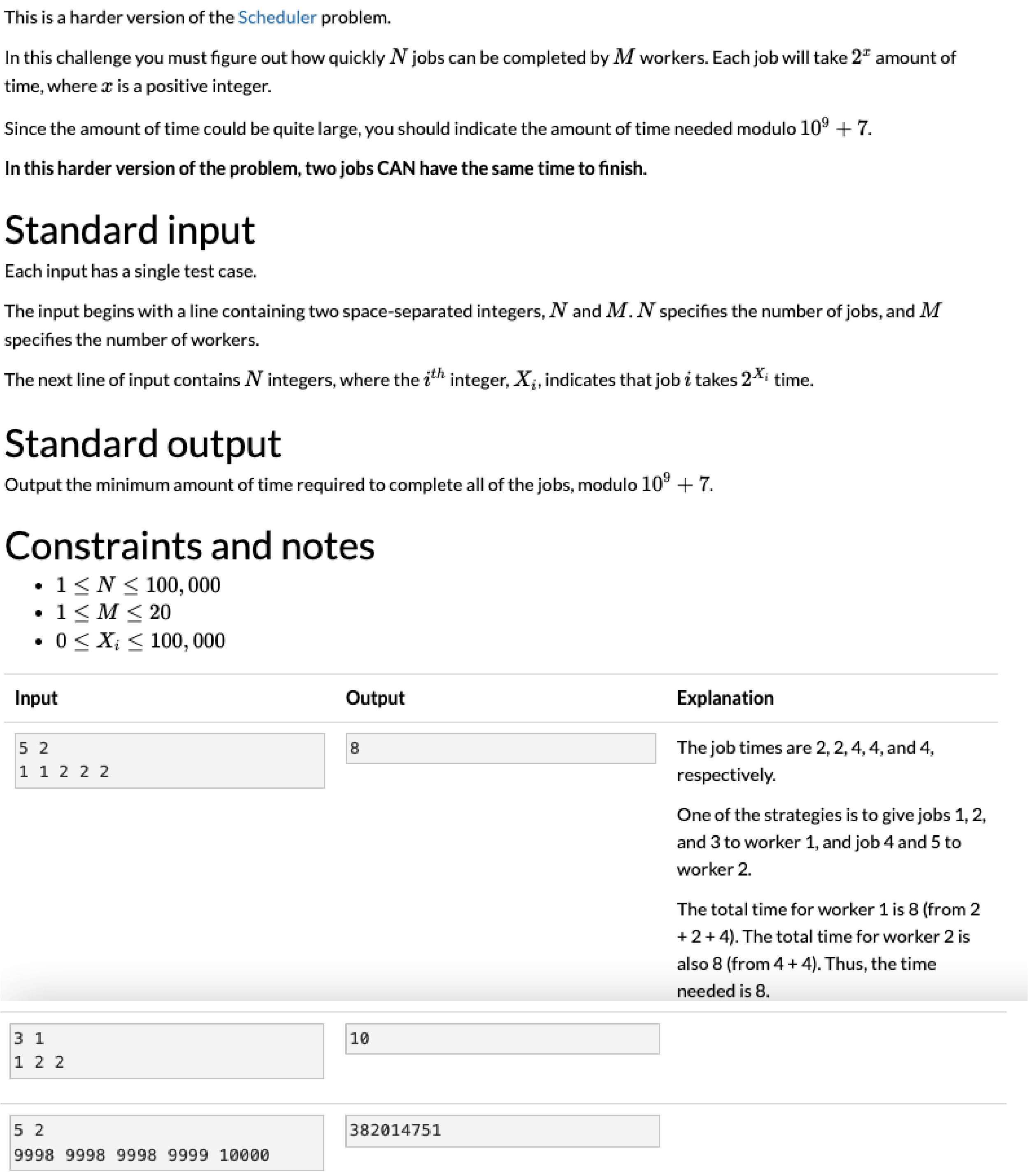
Input	Output	Explanation

`4 3`
`0 1 2 4`
	`16`	The job times, respectively, are 1, 2, 4, 16.
		One strategy is to give job 1 and 3 to Worker 1, Job 2 to Worker 2, and Job 4 to Worker 3. Worker 1 would complete the jobs in 5 time units (1 + 4). Worker 2 would complete the job in 2 time units, and worker 3 would complete the job in 16 units. So, the time needed is 16.

`2 1`
`1 2`
	`6`	

`5 5`
`1 2 4 9999 10000`
	`905611805`	

Scheduler Redux



This is a harder version of the Scheduler problem.

In this challenge you must figure out how quickly N jobs can be completed by M workers. Each job will take 2^x amount of time, where x is a positive integer.

Since the amount of time could be quite large, you should indicate the amount of time needed modulo $10^9 + 7$.

In this harder version of the problem, two jobs CAN have the same time to finish.

Standard input

Each input has a single test case.

The input begins with a line containing two space-separated integers, N and M. N specifies the number of jobs, and M specifies the number of workers.

The next line of input contains N integers, where the i^{th} integer, X_i, indicates that job i takes 2^{X_i} time.

Standard output

Output the minimum amount of time required to complete all of the jobs, modulo $10^9 + 7$.

Constraints and notes

- $1 \leq N \leq 100,000$
- $1 \leq M \leq 20$
- $0 \leq X_i \leq 100,000$

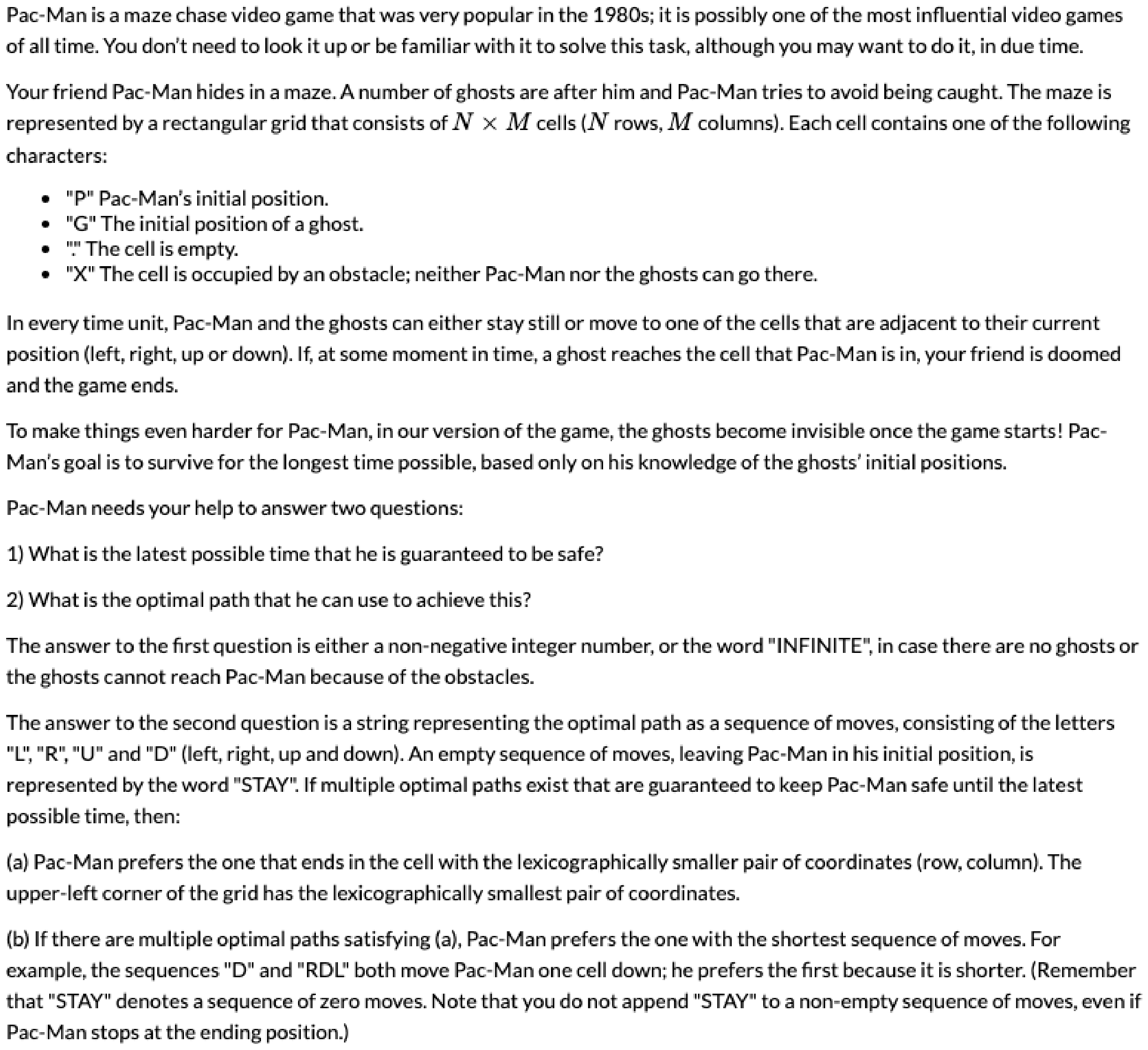
Input	Output	Explanation

`5 2`
`1 1 2 2 2`
	`8`	The job times are 2, 2, 4, 4, and 4, respectively.
		One of the strategies is to give jobs 1, 2, and 3 to worker 1, and job 4 and 5 to worker 2.
		The total time for worker 1 is 8 (from 2 + 2 + 4). The total time for worker 2 is also 8 (from 4 + 4). Thus, the time needed is 8.

`3 1`
`1 2 2`
	`10`	

`5 2`
`9998 9998 9998 9999 10000`
	`382014751`	

Invisible Ghosts



Pac-Man is a maze chase video game that was very popular in the 1980s; it is possibly one of the most influential video games of all time. You don't need to look it up or be familiar with it to solve this task, although you may want to do it, in due time.

Your friend Pac-Man hides in a maze. A number of ghosts are after him and Pac-Man tries to avoid being caught. The maze is represented by a rectangular grid that consists of $N \times M$ cells (N rows, M columns). Each cell contains one of the following characters:

- "P" Pac-Man's initial position.
- "G" The initial position of a ghost.
- "." The cell is empty.
- "X" The cell is occupied by an obstacle; neither Pac-Man nor the ghosts can go there.

In every time unit, Pac-Man and the ghosts can either stay still or move to one of the cells that are adjacent to their current position (left, right, up or down). If, at some moment in time, a ghost reaches the cell that Pac-Man is in, your friend is doomed and the game ends.

To make things even harder for Pac-Man, in our version of the game, the ghosts become invisible once the game starts! Pac-Man's goal is to survive for the longest time possible, based only on his knowledge of the ghosts' initial positions.

Pac-Man needs your help to answer two questions:

1) What is the latest possible time that he is guaranteed to be safe?

2) What is the optimal path that he can use to achieve this?

The answer to the first question is either a non-negative integer number, or the word "INFINITE", in case there are no ghosts or the ghosts cannot reach Pac-Man because of the obstacles.

The answer to the second question is a string representing the optimal path as a sequence of moves, consisting of the letters "L", "R", "U" and "D" (left, right, up and down). An empty sequence of moves, leaving Pac-Man in his initial position, is represented by the word "STAY". If multiple optimal paths exist that are guaranteed to keep Pac-Man safe until the latest possible time, then:

(a) Pac-Man prefers the one that ends in the cell with the lexicographically smaller pair of coordinates (row, column). The upper-left corner of the grid has the lexicographically smallest pair of coordinates.

(b) If there are multiple optimal paths satisfying (a), Pac-Man prefers the one with the shortest sequence of moves. For example, the sequences "D" and "RDL" both move Pac-Man one cell down; he prefers the first because it is shorter. (Remember that "STAY" denotes a sequence of zero moves. Note that you do not append "STAY" to a non-empty sequence of moves, even if Pac-Man stops at the ending position.)

(c) If there are multiple optimal paths satisfying both (a) and (b), Pac-Man prefers the lexicographically smallest sequence of moves. For example, all three sequences "DDR", "DRD" and "RDD" move Pac-Man two cells down and one cell to the right and they have the same length; he prefers "DDR", which is the lexicographically smallest of the three.

Standard input

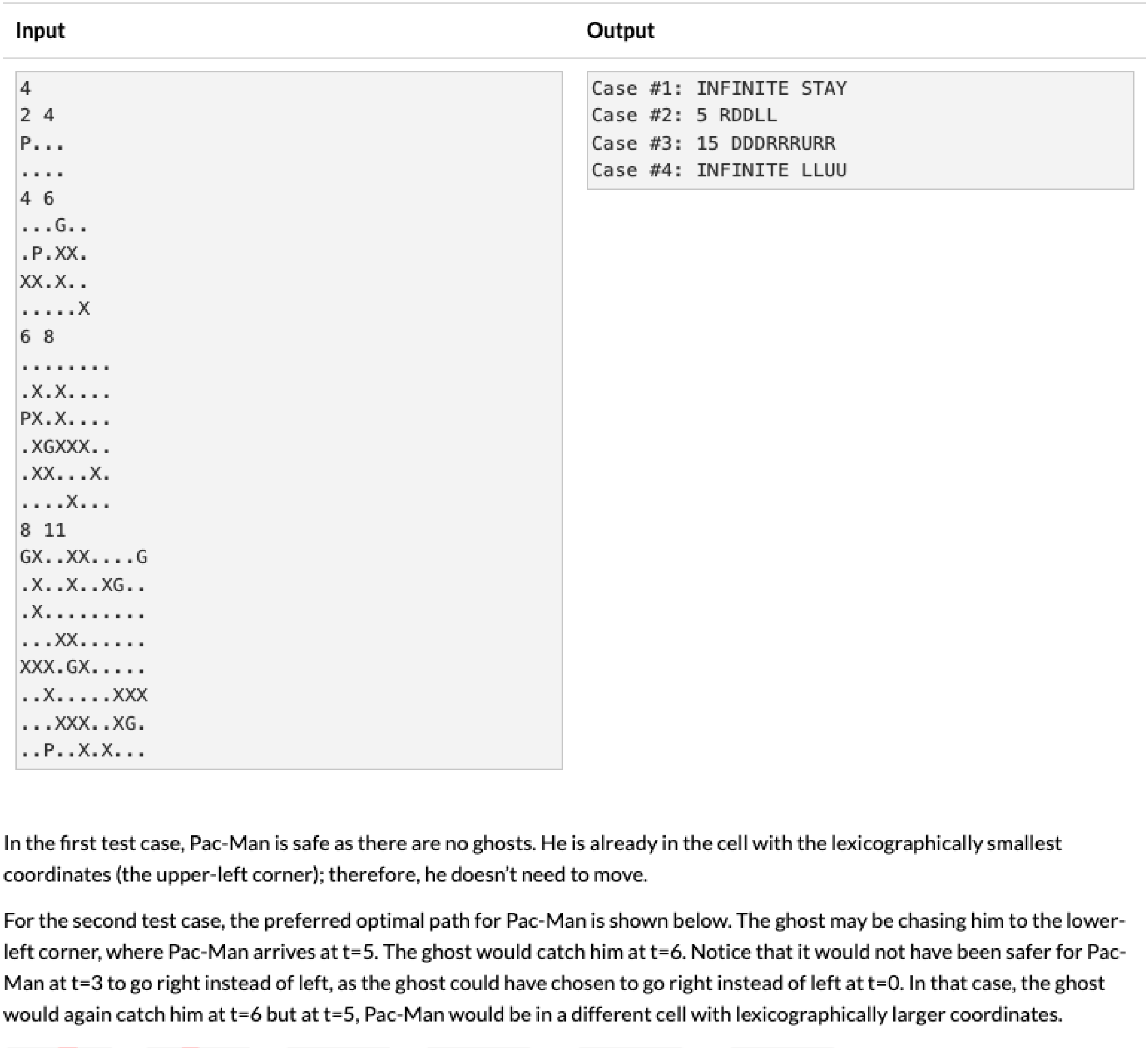
The first line of the input contains an integer T: the number of test cases that follow. Each of the test cases begins with a line containing two integers, N and M, separated by a single space: the dimensions of this test case's grid. N lines follow, each containing M characters among the ones listed above and representing one row of the grid.

Standard output

Your program must print T lines, one for each test case, in the order they are given. The line corresponding to the k^{th} test case ($1 \leq k \leq T$) should contain: "Case #k: ANSWER1 ANSWER2", where ANSWER1 and ANSWER2 are the answers to the two questions, as described above.

Constraints and notes

- $1 \leq T \leq 10$
- $1 \leq \mathrm{N} \leq 500, 1 \leq \mathrm{M} \leq 500$



Input

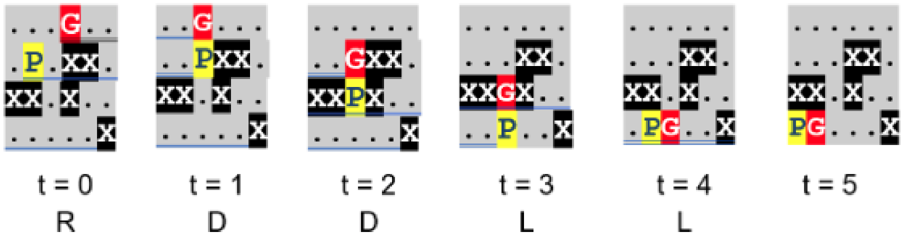
```
4
2 4
P...
....
4 6
...G..
.P.XX.
XX.X..
.....X
6 8
........
.X.X....
PX.X....
.XGXXX..
.XX...X.
....X...
8 11
GX..XX....G
.X..X..XG..
.X.........
...XX......
XXX.GX.....
..X.....XXX
...XXX..XG.
..P..X.X...
```

Output

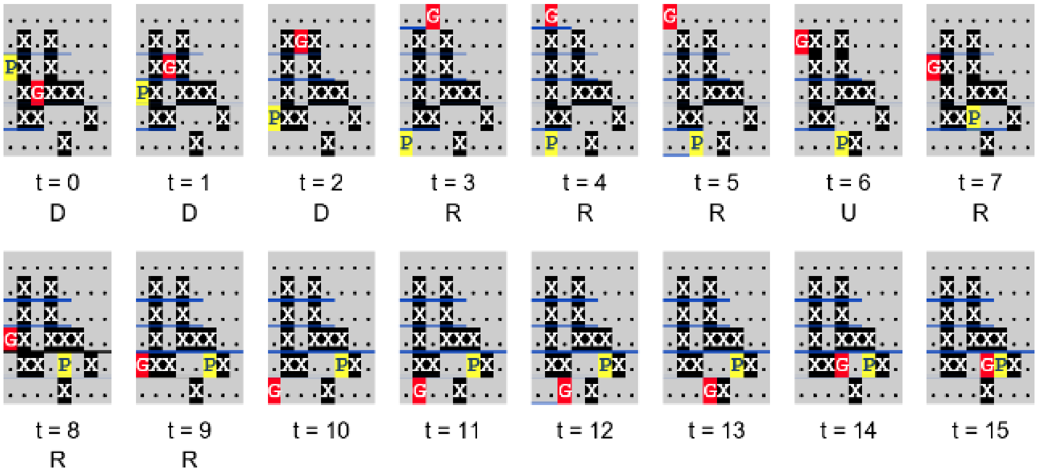
```
Case #1: INFINITE STAY
Case #2: 5 RDDLL
Case #3: 15 DDDRRRURR
Case #4: INFINITE LLUU
```

In the first test case, Pac-Man is safe as there are no ghosts. He is already in the cell with the lexicographically smallest coordinates (the upper-left corner); therefore, he doesn't need to move.

For the second test case, the preferred optimal path for Pac-Man is shown below. The ghost may be chasing him to the lower-left corner, where Pac-Man arrives at t=5. The ghost would catch him at t=6. Notice that it would not have been safer for Pac-Man at t=3 to go right instead of left, as the ghost could have chosen to go right instead of left at t=0. In that case, the ghost would again catch him at t=6 but at t=5, Pac-Man would be in a different cell with lexicographically larger coordinates.



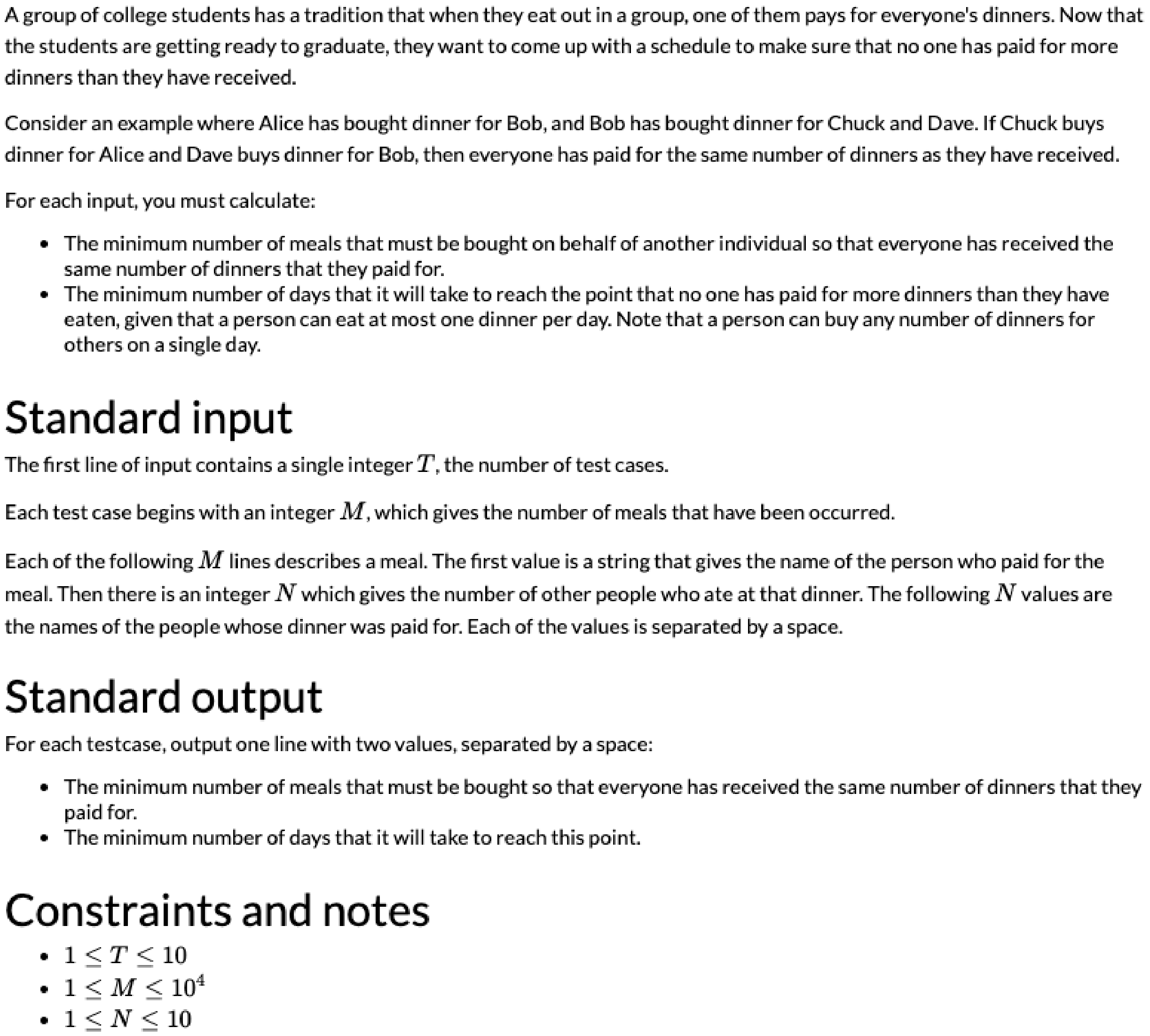
For the third and most interesting test case, the preferred optimal path is again shown below.



Notice that the ghost may reach Pac-Man's final position at t=16. If the ghost had chosen to go right instead of left at t=3, it would again reach the final position at t=16. Therefore, Pac-Man is safe there until t=15 in both cases and, not being able to see the ghost's movement, this is the longest he is guaranteed to stay safe.

In the fourth test case, Pac-Man is safe because the ghosts cannot reach him. He should move to the cell with the lexicographically smallest coordinates that he can reach, following the sequence of four moves "LLUU" (he could also get there with "ULUL" but "LLUU" is lexicographically smaller).

My Treat



A group of college students has a tradition that when they eat out in a group, one of them pays for everyone's dinners. Now that the students are getting ready to graduate, they want to come up with a schedule to make sure that no one has paid for more dinners than they have received.

Consider an example where Alice has bought dinner for Bob, and Bob has bought dinner for Chuck and Dave. If Chuck buys dinner for Alice and Dave buys dinner for Bob, then everyone has paid for the same number of dinners as they have received.

For each input, you must calculate:

- The minimum number of meals that must be bought on behalf of another individual so that everyone has received the same number of dinners that they paid for.
- The minimum number of days that it will take to reach the point that no one has paid for more dinners than they have eaten, given that a person can eat at most one dinner per day. Note that a person can buy any number of dinners for others on a single day.

Standard input

The first line of input contains a single integer T, the number of test cases.

Each test case begins with an integer M, which gives the number of meals that have been occurred.

Each of the following M lines describes a meal. The first value is a string that gives the name of the person who paid for the meal. Then there is an integer N which gives the number of other people who ate at that dinner. The following N values are the names of the people whose dinner was paid for. Each of the values is separated by a space.

Standard output

For each testcase, output one line with two values, separated by a space:

- The minimum number of meals that must be bought so that everyone has received the same number of dinners that they paid for.
- The minimum number of days that it will take to reach this point.

Constraints and notes

- $1 \leq T \leq 10$
- $1 \leq M \leq 10^4$
- $1 \leq N \leq 10$

Constraints and notes

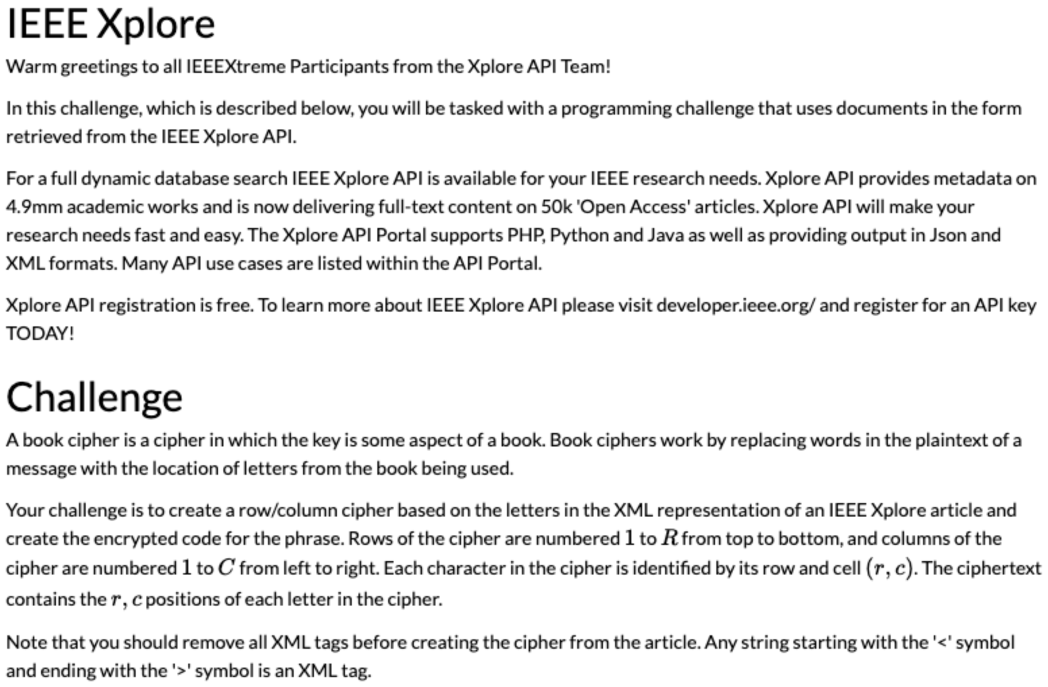
- $1 \leq T \leq 10$
- $1 \leq M \leq 10^4$
- $1 \leq N \leq 10$

Each name will be a string made up of only upper and lower-case letters, up to 20 characters long.

Input	Output	Explanation

`2`
`3`
`Alice 1 Bob`
`Chuck 1 Dave`
`Dave 1 Eve`
`4`
`Alice 2 Bob Chuck`
`Chuck 1 Bob`
`Dave 2 Alice Bob`
`Alice 1 Eve`
	`2 1`
	`4 2`
		In the sample, there are two testcases.
		In the first testcase, Alice bought Bob a dinner, Chuck bought Dave a dinner, and Dave bought Eve a dinner.
		We can achieve the objectives in a single day if Eve buys Chuck dinner and Bob buys Alice dinner. Note that it would also be ok if Eve buys Alice dinner and Bob buys Chuck dinner.
		In the second test case, Alice bought Bob and Chuck dinner, Chuck bought Bob dinner, Dave bought Alice and Bob dinner, and Alice bought Eve dinner. We can achieve zero balances with the following schedule:
		• Day 1: Bob buys dinner for Alice and Dave.
		• Day 2: Bob buys dinner for Alice, and Eve buys dinner for Dave.

Book Cipher



IEEE Xplore

Warm greetings to all IEEEXtreme Participants from the Xplore API Team!

In this challenge, which is described below, you will be tasked with a programming challenge that uses documents in the form retrieved from the IEEE Xplore API.

For a full dynamic database search IEEE Xplore API is available for your IEEE research needs. Xplore API provides metadata on 4.9mm academic works and is now delivering full-text content on 50k 'Open Access' articles. Xplore API will make your research needs fast and easy. The Xplore API Portal supports PHP, Python and Java as well as providing output in Json and XML formats. Many API use cases are listed within the API Portal.

Xplore API registration is free. To learn more about IEEE Xplore API please visit developer.ieee.org/ and register for an API key TODAY!

Challenge

A book cipher is a cipher in which the key is some aspect of a book. Book ciphers work by replacing words in the plaintext of a message with the location of letters from the book being used.

Your challenge is to create a row/column cipher based on the letters in the XML representation of an IEEE Xplore article and create the encrypted code for the phrase. Rows of the cipher are numbered 1 to R from top to bottom, and columns of the cipher are numbered 1 to C from left to right. Each character in the cipher is identified by its row and cell (r, c). The ciphertext contains the r, c positions of each letter in the cipher.

Note that you should remove all XML tags before creating the cipher from the article. Any string starting with the '<' symbol and ending with the '>' symbol is an XML tag.

Standard input

The first line of the input contains an integer p, giving the number of phrases to be encrypted.

The second line of input contains an integer n, indicating the number of lines in the XML representation of Xplore Article.

The third line of input has two integers R, C which give the number of rows and columns, respectively, in the cipher grid. You will fill this grid with the content of the Xplore document. The grid may not be large enough to contain the entire document.

The fourth line of input contains a character that indicates the position code in the cipher for each character. Since a character can appear in the cipher grid more than once you must select the appropriate instance. S denotes finding the lexicographically smallest (R_i, C_i) instance of the character in the grid and L denotes finding the lexicographically largest (R_i, C_i) instance of the character.

The next p lines of the testcase give the phrases to be encrypted.

The final n lines of the testcase are the XML representation of the Xplore document.

Standard output

For each of the p phrases to be encrypted, output either:

- The encoded message in the form $R_1, C_1, R_2, C_2, \ldots, R_z, C_z$

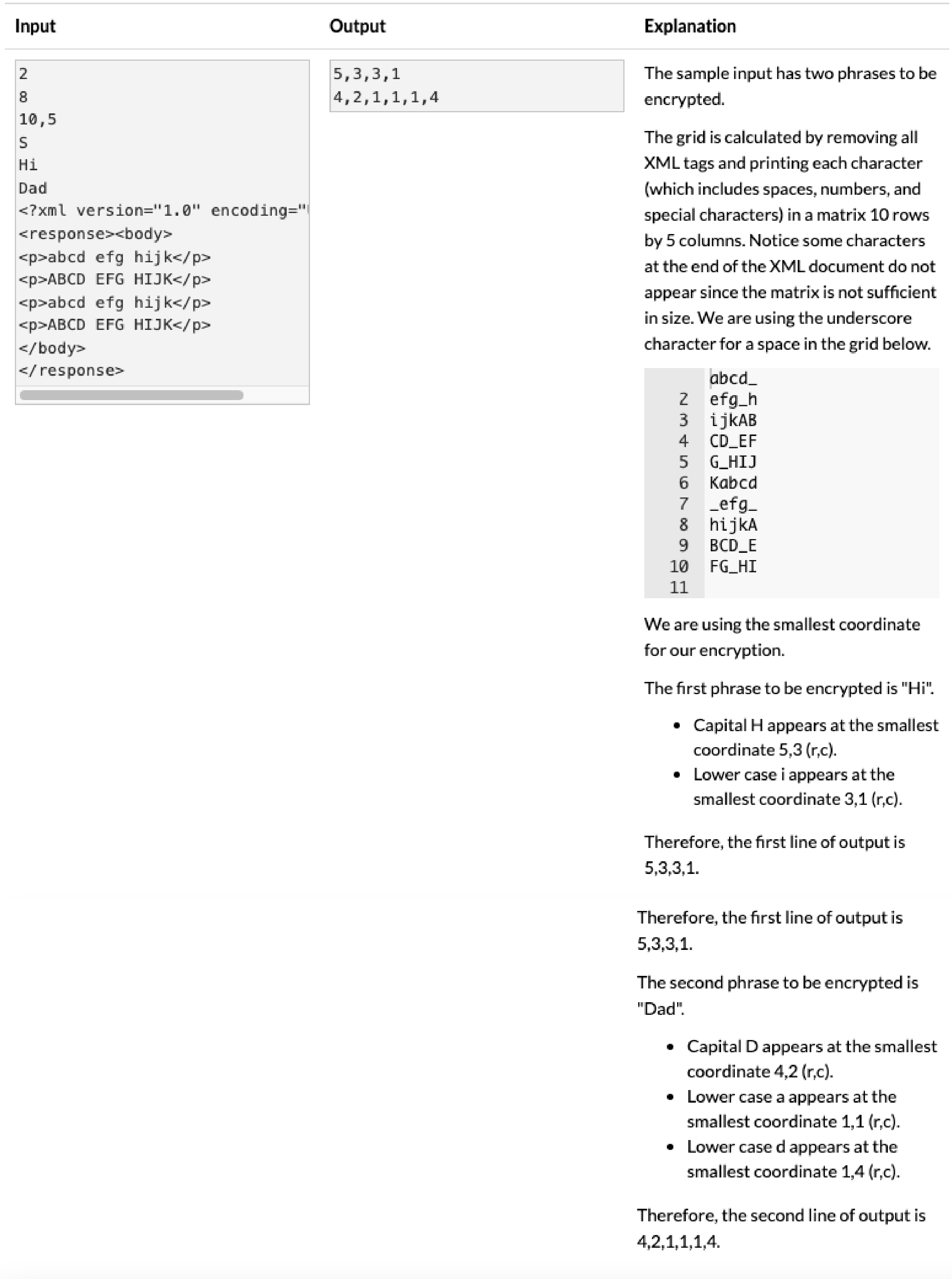
- 0, if the ciphertext cannot be calculated using the cipher.

Constraints and notes

- $1 \leq p \leq 100$
- $1 \leq R, C \leq 2000$
- $1 \leq n \leq 1000$

Each phrase to be encrypted will be between 1 and 1300 characters.

Each input file will be less than 250KB in size.



Input

```
2
8
10,5
S
Hi
Dad
<?xml version="1.0" encoding="
<response><body>
<p>abcd efg hijk</p>
<p>ABCD EFG HIJK</p>
<p>abcd efg hijk</p>
<p>ABCD EFG HIJK</p>
</body>
</response>
```

Output

```
5,3,3,1
4,2,1,1,1,4
```

Explanation

The sample input has two phrases to be encrypted.

The grid is calculated by removing all XML tags and printing each character (which includes spaces, numbers, and special characters) in a matrix 10 rows by 5 columns. Notice some characters at the end of the XML document do not appear since the matrix is not sufficient in size. We are using the underscore character for a space in the grid below.

```
abcd_
efg_h
ijkAB
CD_EF
G_HIJ
Kabcd
_efg_
hijkA
BCD_E
FG_HI
```

We are using the smallest coordinate for our encryption.

The first phrase to be encrypted is "Hi".

- Capital H appears at the smallest coordinate 5,3 (r,c).
- Lower case i appears at the smallest coordinate 3,1 (r,c).

Therefore, the first line of output is 5,3,3,1.

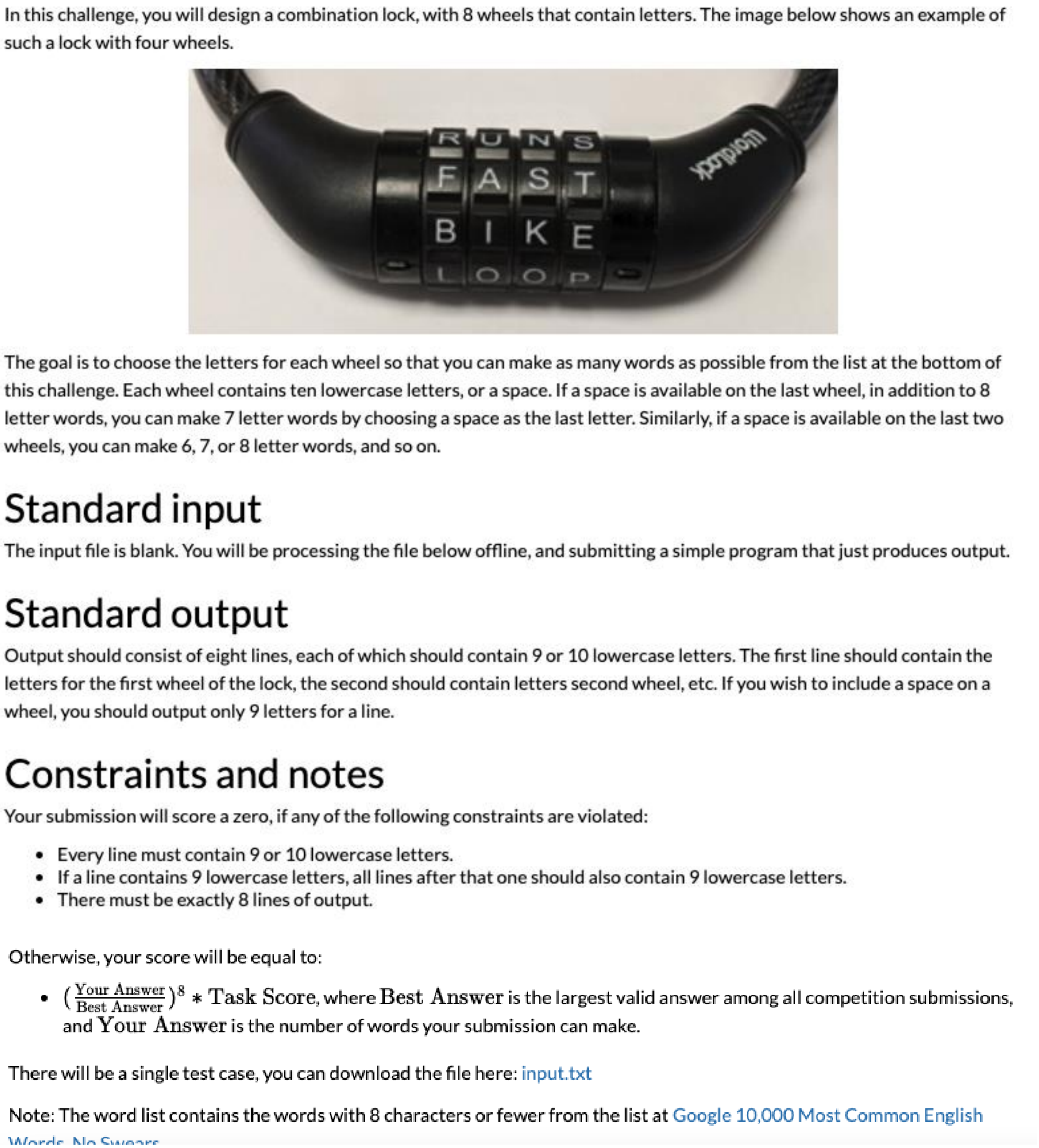
Therefore, the first line of output is 5,3,3,1.

The second phrase to be encrypted is "Dad".

- Capital D appears at the smallest coordinate 4,2 (r,c).
- Lower case a appears at the smallest coordinate 1,1 (r,c).
- Lower case d appears at the smallest coordinate 1,4 (r,c).

Therefore, the second line of output is 4,2,1,1,1,4.

Combo Locks



In this challenge, you will design a combination lock, with 8 wheels that contain letters. The image below shows an example of such a lock with four wheels.

The goal is to choose the letters for each wheel so that you can make as many words as possible from the list at the bottom of this challenge. Each wheel contains ten lowercase letters, or a space. If a space is available on the last wheel, in addition to 8 letter words, you can make 7 letter words by choosing a space as the last letter. Similarly, if a space is available on the last two wheels, you can make 6, 7, or 8 letter words, and so on.

Standard input

The input file is blank. You will be processing the file below offline, and submitting a simple program that just produces output.

Standard output

Output should consist of eight lines, each of which should contain 9 or 10 lowercase letters. The first line should contain the letters for the first wheel of the lock, the second should contain letters second wheel, etc. If you wish to include a space on a wheel, you should output only 9 letters for a line.

Constraints and notes

Your submission will score a zero, if any of the following constraints are violated:

- Every line must contain 9 or 10 lowercase letters.
- If a line contains 9 lowercase letters, all lines after that one should also contain 9 lowercase letters.
- There must be exactly 8 lines of output.

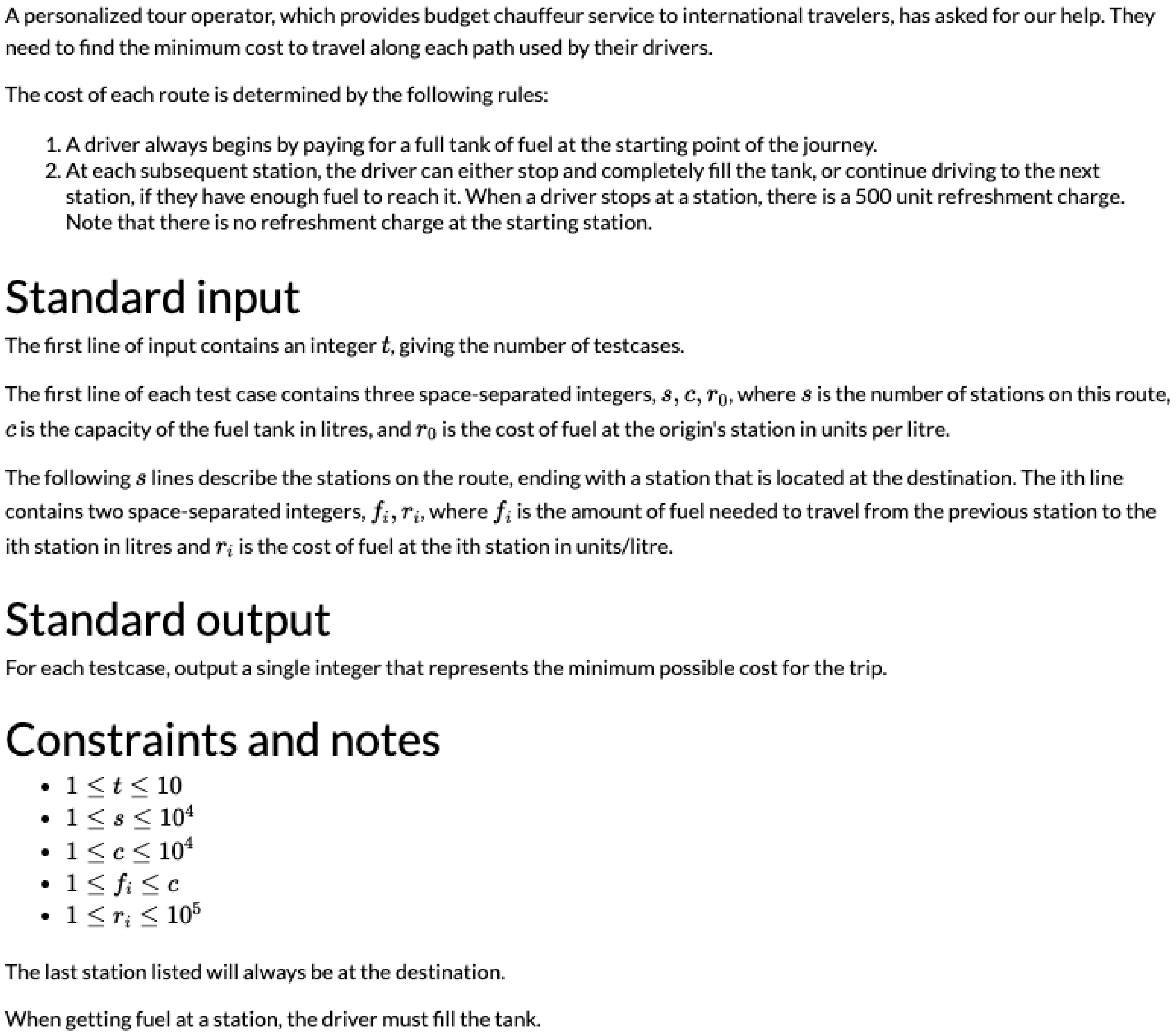
Otherwise, your score will be equal to:

- $(\frac{\text{Your Answer}}{\text{Best Answer}})^8 * \text{Task Score}$, where Best Answer is the largest valid answer among all competition submissions, and Your Answer is the number of words your submission can make.

There will be a single test case, you can download the file here: input.txt

Note: The word list contains the words with 8 characters or fewer from the list at Google 10,000 Most Common English Words, No Swears

Travel Service



A personalized tour operator, which provides budget chauffeur service to international travelers, has asked for our help. They need to find the minimum cost to travel along each path used by their drivers.

The cost of each route is determined by the following rules:

1. A driver always begins by paying for a full tank of fuel at the starting point of the journey.
2. At each subsequent station, the driver can either stop and completely fill the tank, or continue driving to the next station, if they have enough fuel to reach it. When a driver stops at a station, there is a 500 unit refreshment charge. Note that there is no refreshment charge at the starting station.

Standard input

The first line of input contains an integer t, giving the number of testcases.

The first line of each test case contains three space-separated integers, s, c, r_0, where s is the number of stations on this route, c is the capacity of the fuel tank in litres, and r_0 is the cost of fuel at the origin's station in units per litre.

The following s lines describe the stations on the route, ending with a station that is located at the destination. The ith line contains two space-separated integers, f_i, r_i, where f_i is the amount of fuel needed to travel from the previous station to the ith station in litres and r_i is the cost of fuel at the ith station in units/litre.

Standard output

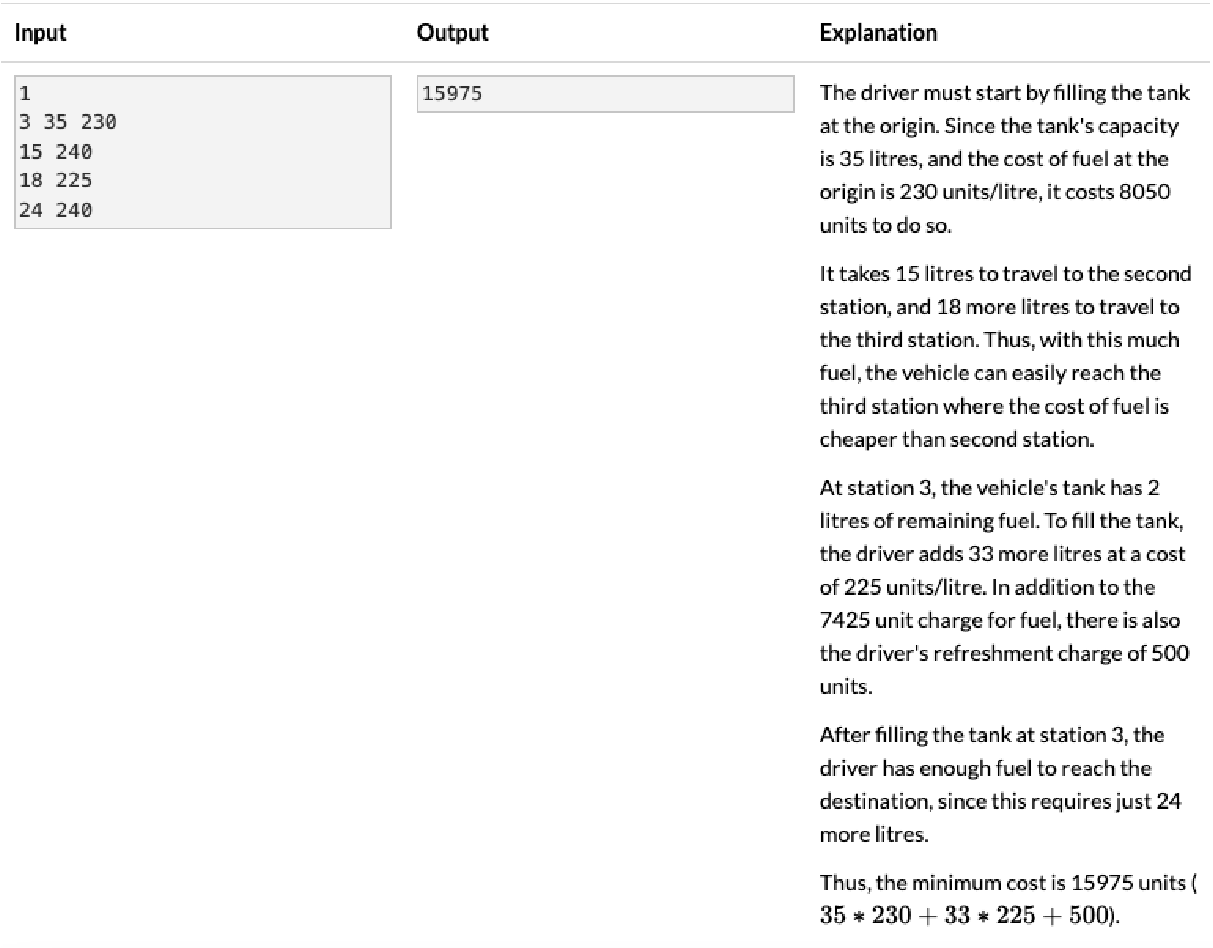
For each testcase, output a single integer that represents the minimum possible cost for the trip.

Constraints and notes

- $1 \leq t \leq 10$
- $1 \leq s \leq 10^4$
- $1 \leq c \leq 10^4$
- $1 \leq f_i \leq c$
- $1 \leq r_i \leq 10^5$

The last station listed will always be at the destination.

When getting fuel at a station, the driver must fill the tank.



Input	Output	Explanation

`1`
`3 35 230`
`15 240`
`18 225`
`24 240`
	`15975`	The driver must start by filling the tank at the origin. Since the tank's capacity is 35 litres, and the cost of fuel at the origin is 230 units/litre, it costs 8050 units to do so.
		It takes 15 litres to travel to the second station, and 18 more litres to travel to the third station. Thus, with this much fuel, the vehicle can easily reach the third station where the cost of fuel is cheaper than second station.
		At station 3, the vehicle's tank has 2 litres of remaining fuel. To fill the tank, the driver adds 33 more litres at a cost of 225 units/litre. In addition to the 7425 unit charge for fuel, there is also the driver's refreshment charge of 500 units.
		After filling the tank at station 3, the driver has enough fuel to reach the destination, since this requires just 24 more litres.
		Thus, the minimum cost is 15975 units ($35 * 230 + 33 * 225 + 500$).

Outpost

During the combat against Zerg, Terran is planning to set up an outpost on the narrow and long bridge between the two sides of Neverwhere Canyon. Your task is to help Terran select the best rectangular area on this natural bridge as the outpost.

The surface of the bridge can be considered as a fat matrix or a 2D-array A, with a size of $m \times n$. As a natural bridge, ups and downs can be observed. Entries of the matrix indicate the altitude. For simplification purposes, the entries are non-negative integers.

An outpost O is in a rectangular area paralleling the edges of the matrix A and can only be established on a relatively flat area. The highest and lowest elevation within the outpost area cannot exceed a given threshold t. In a mathematics statement,

$$max_{(i,j)\in O} A_{ij} - min_{(i,j)\in O} A_{ij} \le t, O = \{(i,j) | x_0 \le i \le x_1, y_0 \le j \le y_1\}$$

In addition, in order to set up sentinels and flags, the perimeter of the outpost must contain a point with maximum elevation in the outpost area.

Terran wants to set up the outpost with the maximum possible area given the above requirements.

Standard input

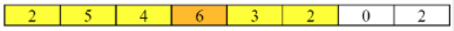
The first row of the input has three integers representing narrow bridge size m, n, and the tolerated maximum altitude difference t. Then, the following m rows each have n integers representing the matrix A, which indicates each point's altitude.

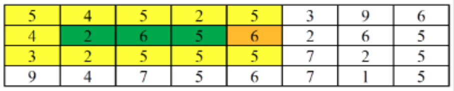
Standard output

An integer for the maximum area of a legal outpost should be the output.

Constraints and notes

- $0 \le A_{ij} \le 10^9$
- $0 \le t \le 10^9$
- For 30% of the test cases, $m = 1$ AND $n \le 50000$.
- In the remaining test cases, $m < n$ and $m \le 10$ and $n \le 5000$.

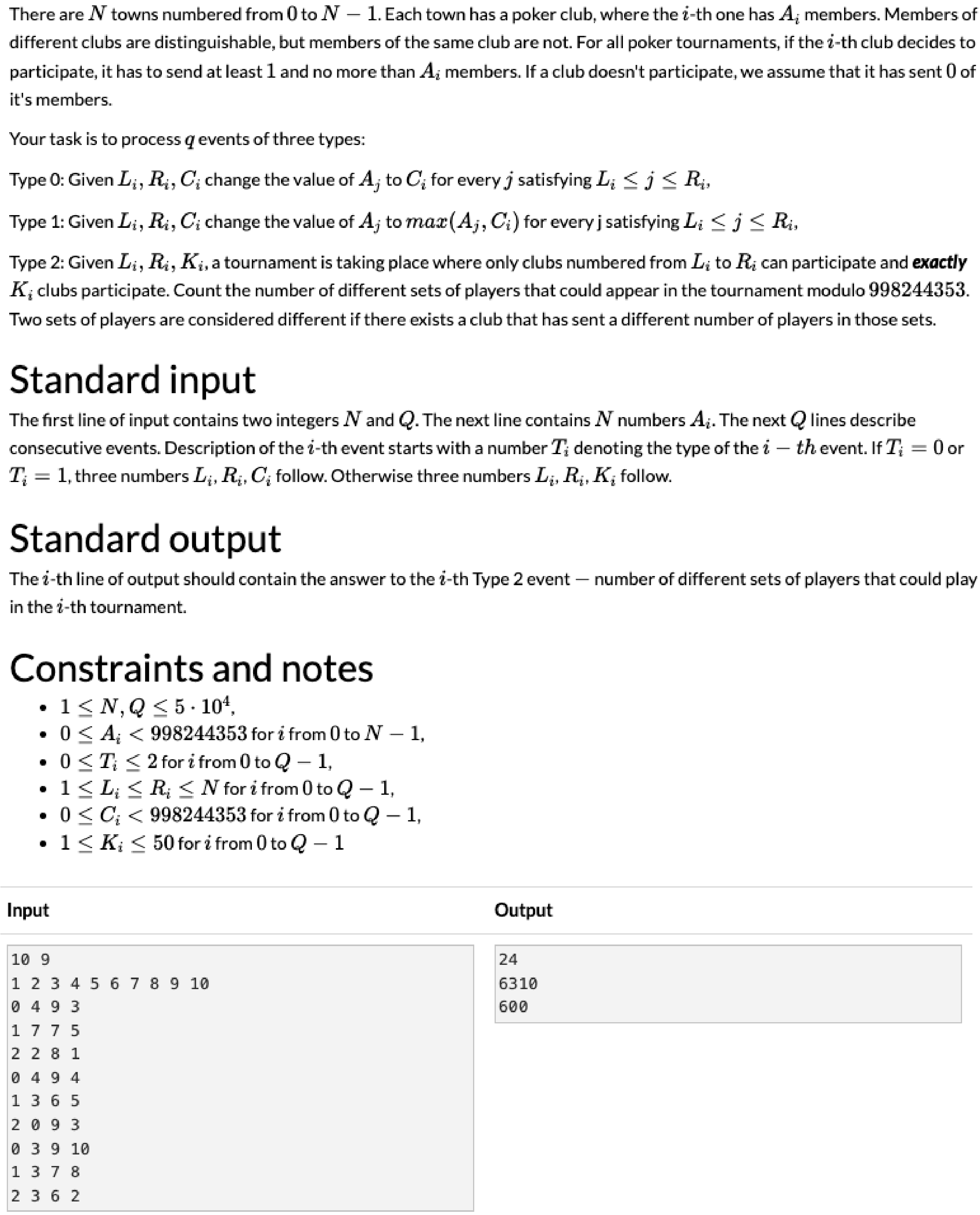
Input	Output	Explanation
1 8 4 2 5 4 6 3 2 0 2	6	The maximum outpost is shown below. The highest (6) minus the lowest (2) is less than or equal to the given threshold (4). Since it is a row when $m = 1$, each entry in the outpost is on the edge of the outpost. 

4 8 4
5 4 5 2 5 3 9 6
4 2 6 5 6 2 6 5
3 2 5 5 5 7 2 5
9 4 7 5 6 7 1 5
	15	The maximum outpost is shown below. The highest (6) minus the lowest (2) is less than or equal to the given threshold (4). Note that the maximum elevation, 6, is present in an entry on the perimeter of the outpost.
		

2	5	4	6	3	2	0	2

5	4	5	2	5	3	9	6
4	2	6	5	6	2	6	5
3	2	5	5	5	7	2	5
9	4	7	5	6	7	1	5

Poker Tournament



There are N towns numbered from 0 to $N-1$. Each town has a poker club, where the i-th one has A_i members. Members of different clubs are distinguishable, but members of the same club are not. For all poker tournaments, if the i-th club decides to participate, it has to send at least 1 and no more than A_i members. If a club doesn't participate, we assume that it has sent 0 of it's members.

Your task is to process q events of three types:

Type 0: Given L_i, R_i, C_i change the value of A_j to C_i for every j satisfying $L_i \leq j \leq R_i$,

Type 1: Given L_i, R_i, C_i change the value of A_j to $max(A_j, C_i)$ for every j satisfying $L_i \leq j \leq R_i$,

Type 2: Given L_i, R_i, K_i, a tournament is taking place where only clubs numbered from L_i to R_i can participate and ***exactly*** K_i clubs participate. Count the number of different sets of players that could appear in the tournament modulo 998244353. Two sets of players are considered different if there exists a club that has sent a different number of players in those sets.

Standard input

The first line of input contains two integers N and Q. The next line contains N numbers A_i. The next Q lines describe consecutive events. Description of the i-th event starts with a number T_i denoting the type of the $i-th$ event. If $T_i = 0$ or $T_i = 1$, three numbers L_i, R_i, C_i follow. Otherwise three numbers L_i, R_i, K_i follow.

Standard output

The i-th line of output should contain the answer to the i-th Type 2 event — number of different sets of players that could play in the i-th tournament.

Constraints and notes

- $1 \leq N, Q \leq 5 \cdot 10^4$,
- $0 \leq A_i < 998244353$ for i from 0 to $N-1$,
- $0 \leq T_i \leq 2$ for i from 0 to $Q-1$,
- $1 \leq L_i \leq R_i \leq N$ for i from 0 to $Q-1$,
- $0 \leq C_i < 998244353$ for i from 0 to $Q-1$,
- $1 \leq K_i \leq 50$ for i from 0 to $Q-1$

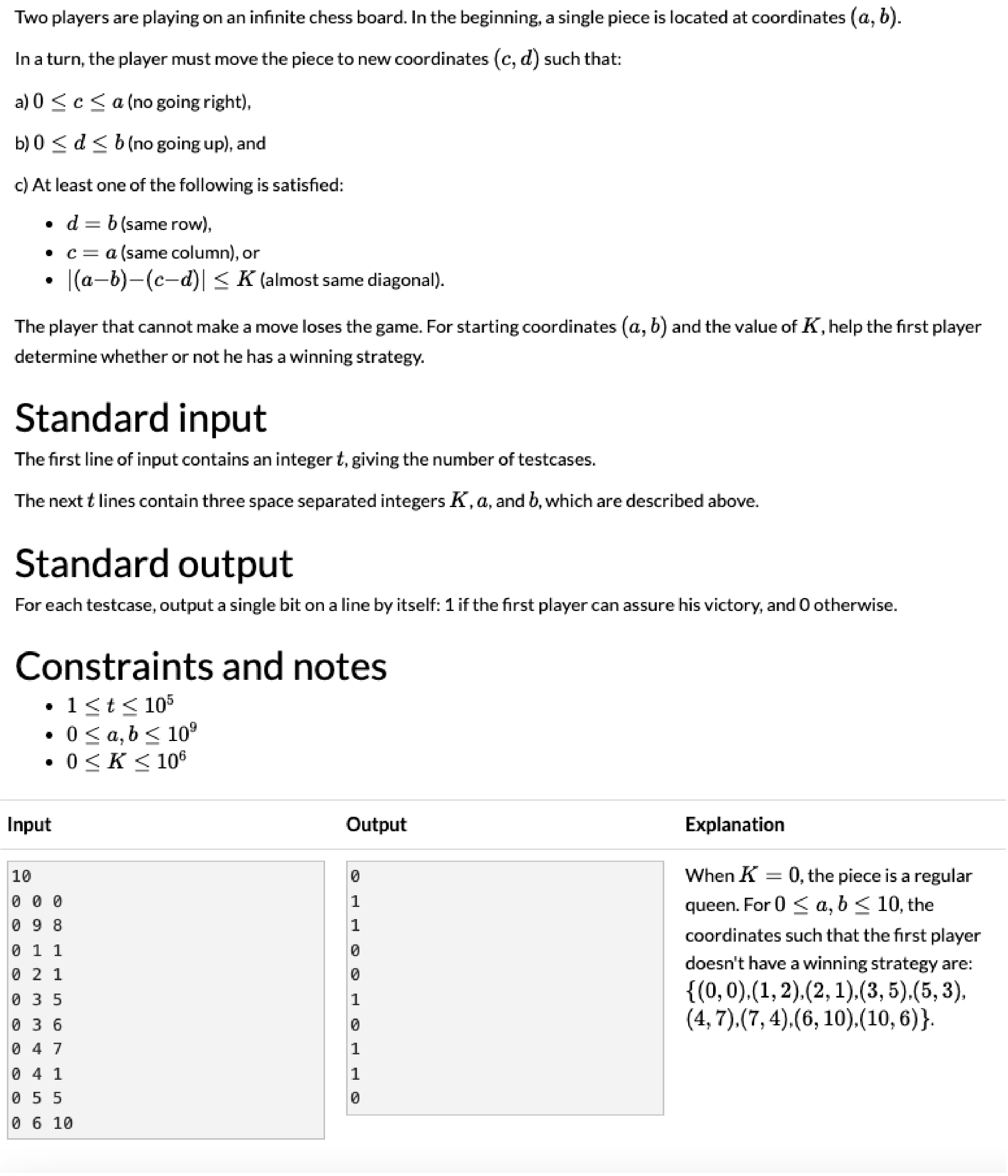
Input

```
10 9
1 2 3 4 5 6 7 8 9 10
0 4 9 3
1 7 7 5
2 2 8 1
0 4 9 4
1 3 6 5
2 0 9 3
0 3 9 10
1 3 7 8
2 3 6 2
```

Output

```
24
6310
600
```

Queen



Two players are playing on an infinite chess board. In the beginning, a single piece is located at coordinates (a, b).

In a turn, the player must move the piece to new coordinates (c, d) such that:

a) $0 \leq c \leq a$ (no going right),

b) $0 \leq d \leq b$ (no going up), and

c) At least one of the following is satisfied:

- $d = b$ (same row),
- $c = a$ (same column), or
- $|(a{-}b){-}(c{-}d)| \leq K$ (almost same diagonal).

The player that cannot make a move loses the game. For starting coordinates (a, b) and the value of K, help the first player determine whether or not he has a winning strategy.

Standard input

The first line of input contains an integer t, giving the number of testcases.

The next t lines contain three space separated integers K, a, and b, which are described above.

Standard output

For each testcase, output a single bit on a line by itself: 1 if the first player can assure his victory, and 0 otherwise.

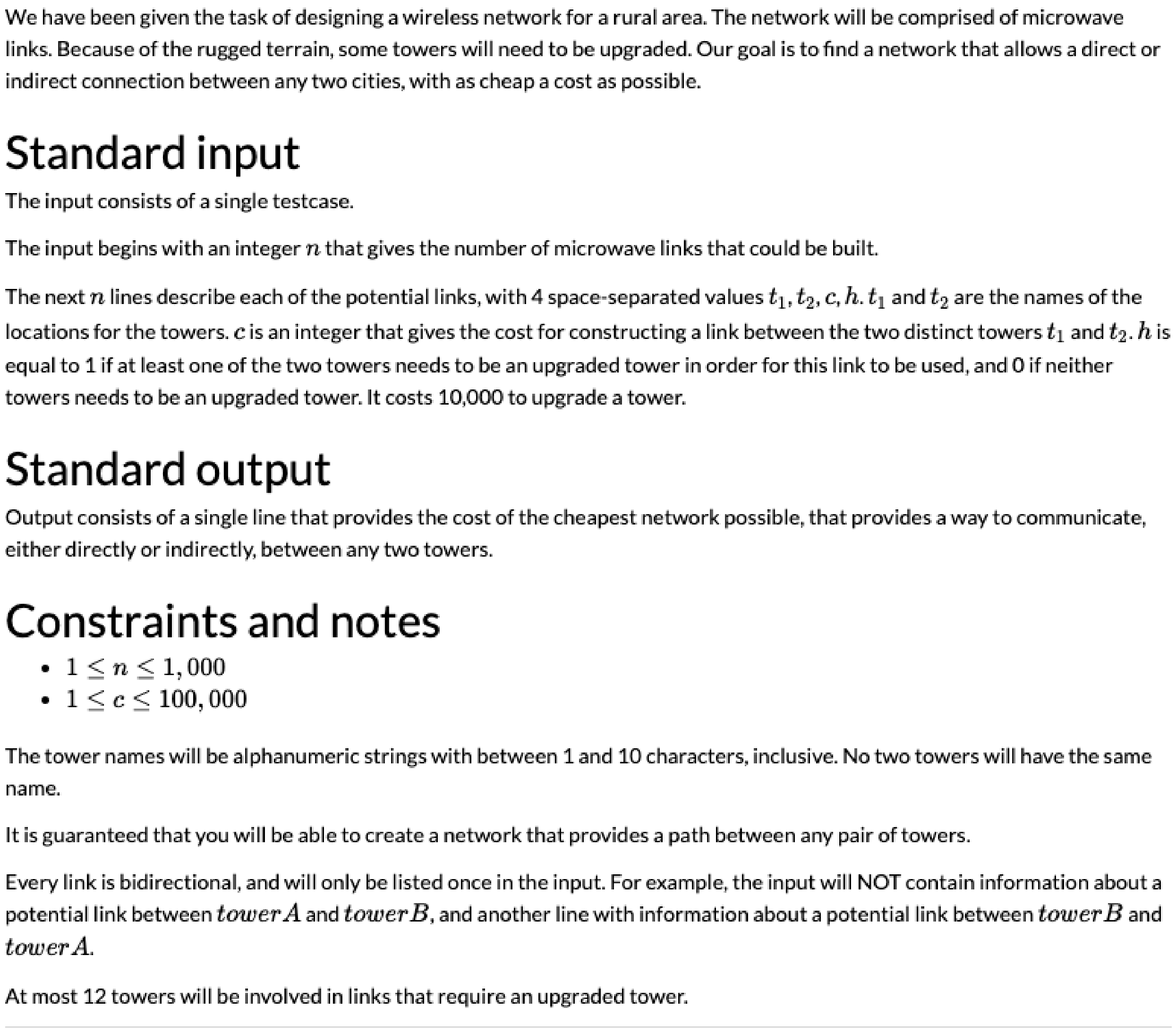
Constraints and notes

- $1 \leq t \leq 10^5$
- $0 \leq a, b \leq 10^9$
- $0 \leq K \leq 10^6$

Input	Output	Explanation

10
0 0 0
0 9 8
0 1 1
0 2 1
0 3 5
0 3 6
0 4 7
0 4 1
0 5 5
0 6 10
	0
	1
	1
	0
	0
	1
	0
	1
	1
	0
		When $K = 0$, the piece is a regular queen. For $0 \leq a, b \leq 10$, the coordinates such that the first player doesn't have a winning strategy are: $\{(0, 0)$,$(1, 2)$,$(2, 1)$,$(3, 5)$,$(5, 3)$,$(4, 7)$,$(7, 4)$,$(6, 10)$,$(10, 6)\}$.

Wireless Network



We have been given the task of designing a wireless network for a rural area. The network will be comprised of microwave links. Because of the rugged terrain, some towers will need to be upgraded. Our goal is to find a network that allows a direct or indirect connection between any two cities, with as cheap a cost as possible.

Standard input

The input consists of a single testcase.

The input begins with an integer n that gives the number of microwave links that could be built.

The next n lines describe each of the potential links, with 4 space-separated values t_1, t_2, c, h. t_1 and t_2 are the names of the locations for the towers. c is an integer that gives the cost for constructing a link between the two distinct towers t_1 and t_2. h is equal to 1 if at least one of the two towers needs to be an upgraded tower in order for this link to be used, and 0 if neither towers needs to be an upgraded tower. It costs 10,000 to upgrade a tower.

Standard output

Output consists of a single line that provides the cost of the cheapest network possible, that provides a way to communicate, either directly or indirectly, between any two towers.

Constraints and notes

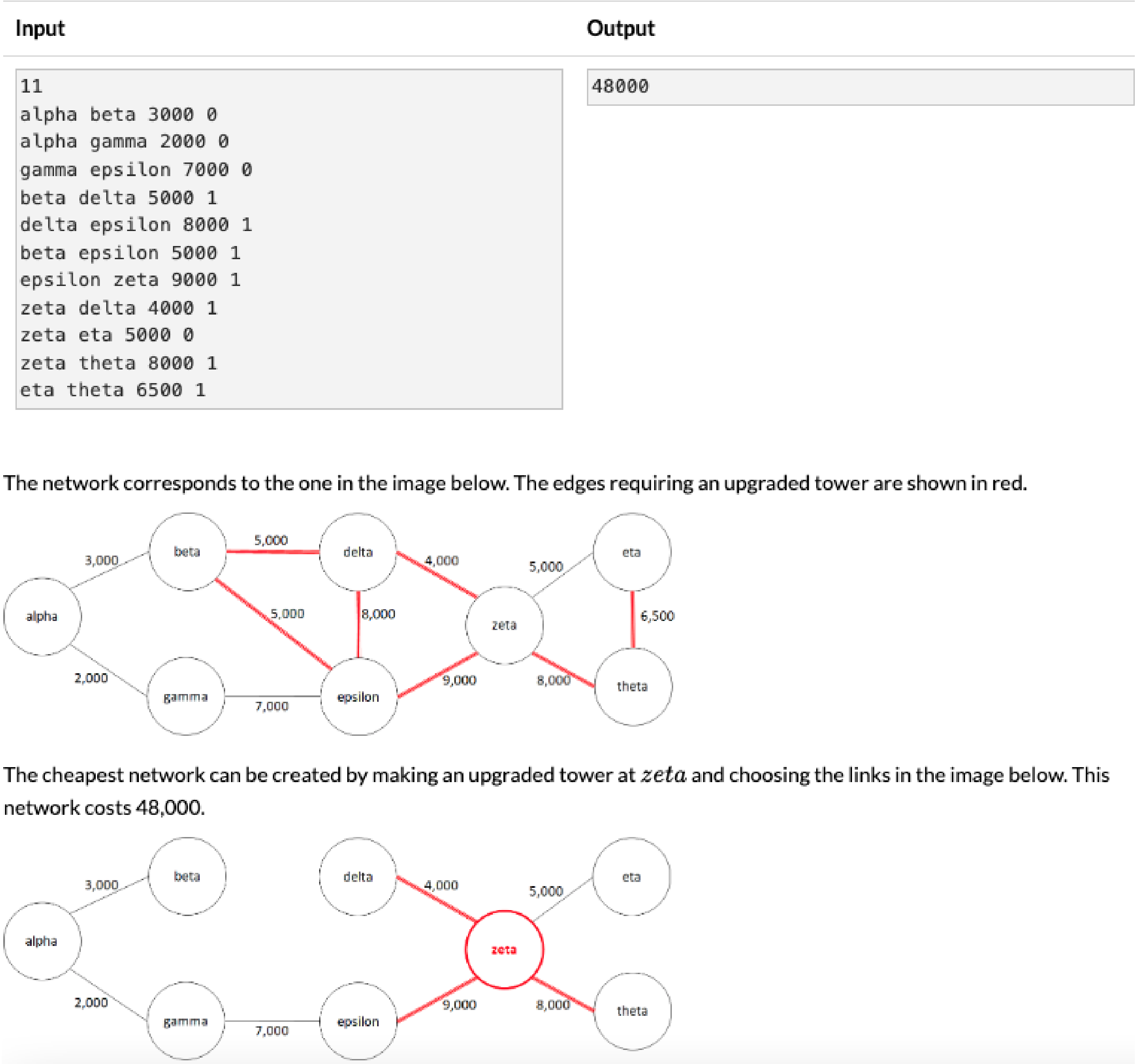
- $1 \leq n \leq 1,000$
- $1 \leq c \leq 100,000$

The tower names will be alphanumeric strings with between 1 and 10 characters, inclusive. No two towers will have the same name.

It is guaranteed that you will be able to create a network that provides a path between any pair of towers.

Every link is bidirectional, and will only be listed once in the input. For example, the input will NOT contain information about a potential link between $towerA$ and $towerB$, and another line with information about a potential link between $towerB$ and $towerA$.

At most 12 towers will be involved in links that require an upgraded tower.



Input

```
11
alpha beta 3000 0
alpha gamma 2000 0
gamma epsilon 7000 0
beta delta 5000 1
delta epsilon 8000 1
beta epsilon 5000 1
epsilon zeta 9000 1
zeta delta 4000 1
zeta eta 5000 0
zeta theta 8000 1
eta theta 6500 1
```

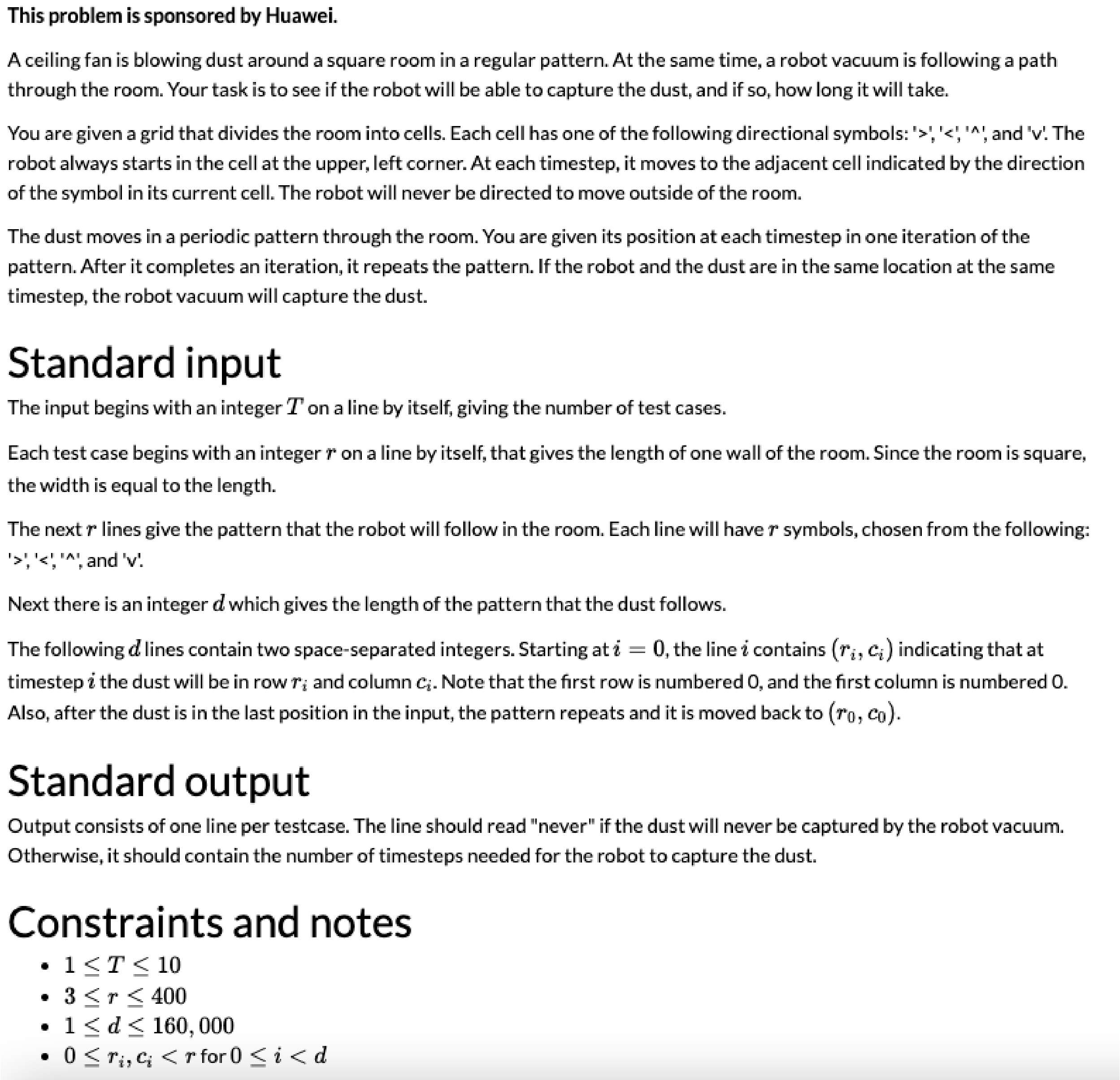
Output

```
48000
```

The network corresponds to the one in the image below. The edges requiring an upgraded tower are shown in red.

The cheapest network can be created by making an upgraded tower at $zeta$ and choosing the links in the image below. This network costs 48,000.

Robot versus Fan



This problem is sponsored by Huawei.

A ceiling fan is blowing dust around a square room in a regular pattern. At the same time, a robot vacuum is following a path through the room. Your task is to see if the robot will be able to capture the dust, and if so, how long it will take.

You are given a grid that divides the room into cells. Each cell has one of the following directional symbols: '>', '<', '^', and 'v'. The robot always starts in the cell at the upper, left corner. At each timestep, it moves to the adjacent cell indicated by the direction of the symbol in its current cell. The robot will never be directed to move outside of the room.

The dust moves in a periodic pattern through the room. You are given its position at each timestep in one iteration of the pattern. After it completes an iteration, it repeats the pattern. If the robot and the dust are in the same location at the same timestep, the robot vacuum will capture the dust.

Standard input

The input begins with an integer T on a line by itself, giving the number of test cases.

Each test case begins with an integer r on a line by itself, that gives the length of one wall of the room. Since the room is square, the width is equal to the length.

The next r lines give the pattern that the robot will follow in the room. Each line will have r symbols, chosen from the following: '>', '<', '^', and 'v'.

Next there is an integer d which gives the length of the pattern that the dust follows.

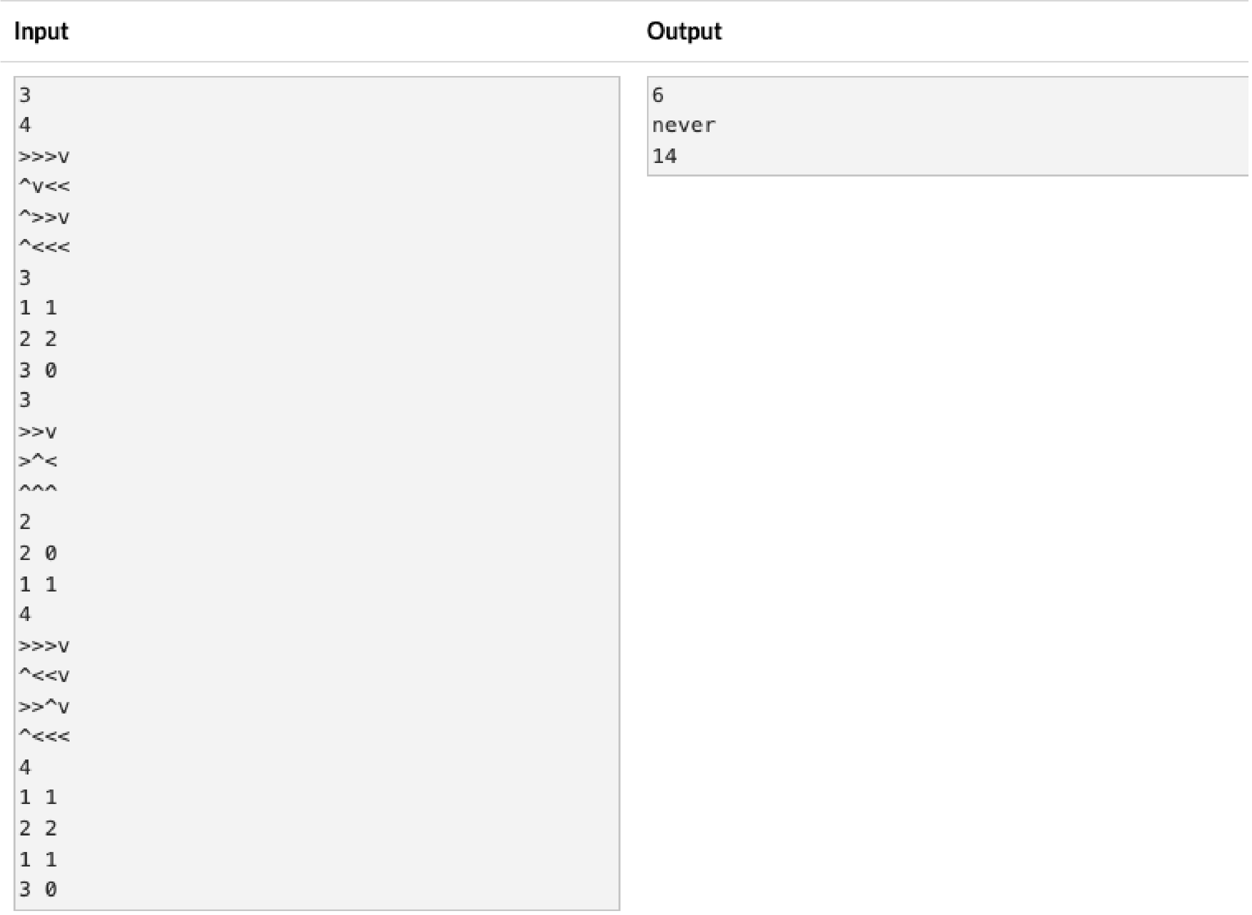
The following d lines contain two space-separated integers. Starting at $i = 0$, the line i contains (r_i, c_i) indicating that at timestep i the dust will be in row r_i and column c_i. Note that the first row is numbered 0, and the first column is numbered 0. Also, after the dust is in the last position in the input, the pattern repeats and it is moved back to (r_0, c_0).

Standard output

Output consists of one line per testcase. The line should read "never" if the dust will never be captured by the robot vacuum. Otherwise, it should contain the number of timesteps needed for the robot to capture the dust.

Constraints and notes

- $1 \leq T \leq 10$
- $3 \leq r \leq 400$
- $1 \leq d \leq 160,000$
- $0 \leq r_i, c_i < r$ for $0 \leq i < d$



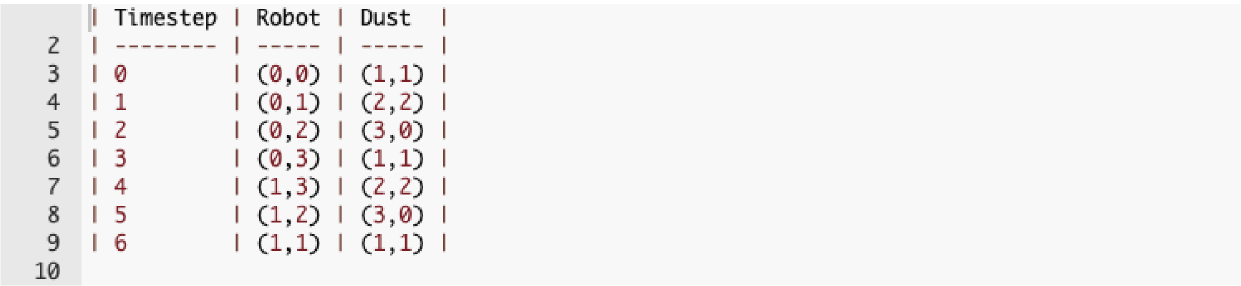
Input

```
3
4
>>>v
^v<<
^>>v
^<<<
3
1 1
2 2
3 0
3
>>v
>^<
^^^
2
2 0
1 1
4
>>>v
^<<v
>>^v
^<<<
4
1 1
2 2
1 1
3 0
```

Output

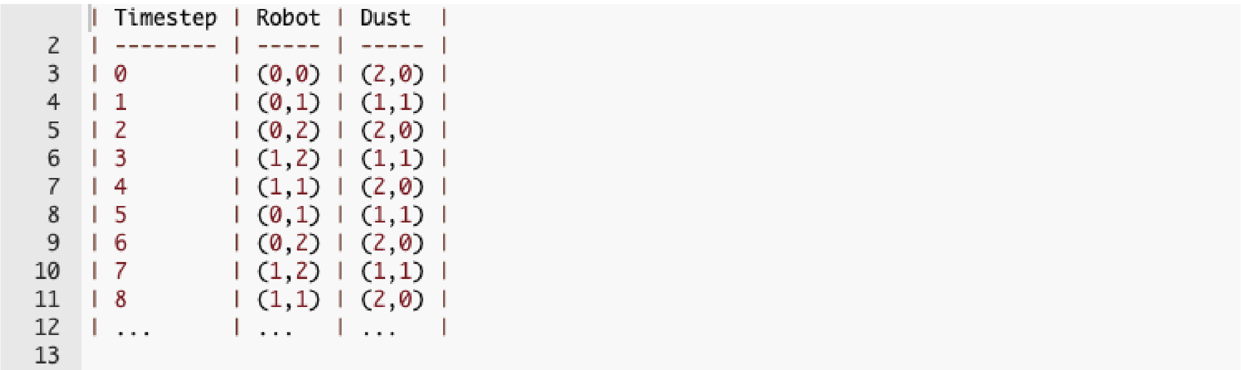
```
6
never
14
```

The table below gives the robot vacuum and dust positions as a (row, column) pair. Note that the dust locations repeat every three timesteps. Since the robot and the dust are in the same position at timestep 6, you would output 6.



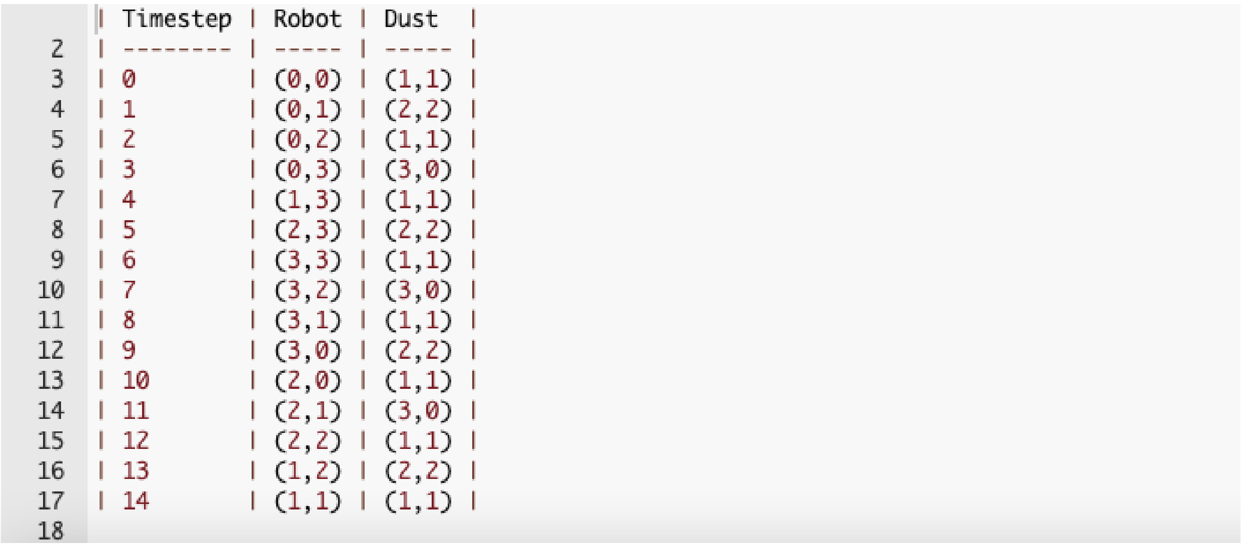
```
| Timestep | Robot | Dust  |
| -------- | ----- | ----- |
| 0        | (0,0) | (1,1) |
| 1        | (0,1) | (2,2) |
| 2        | (0,2) | (3,0) |
| 3        | (0,3) | (1,1) |
| 4        | (1,3) | (2,2) |
| 5        | (1,2) | (3,0) |
| 6        | (1,1) | (1,1) |
```

For the second test case, the first few positions for the robot and the dust are given in the table below. Note that the robot never visits cell (2,0). The robot and the dust both visit (1,1), but in the table, the robot only visits in even timesteps and the dust only visits in odd timesteps. This pattern would repeat forever without the robot and dust ever being in the same cell. Thus, you would output "never".



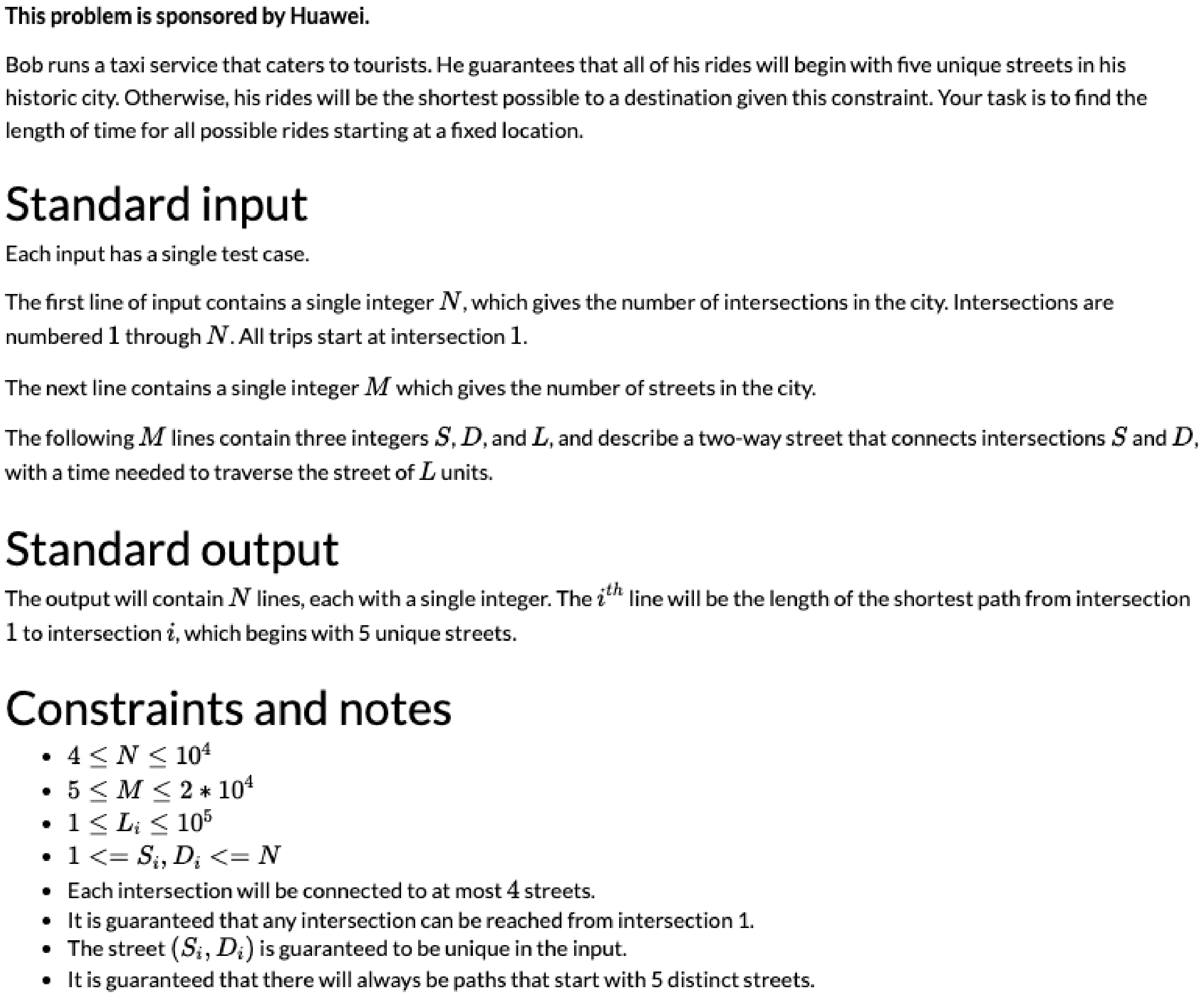
```
| Timestep | Robot | Dust  |
| -------- | ----- | ----- |
| 0        | (0,0) | (2,0) |
| 1        | (0,1) | (1,1) |
| 2        | (0,2) | (2,0) |
| 3        | (1,2) | (1,1) |
| 4        | (1,1) | (2,0) |
| 5        | (0,1) | (1,1) |
| 6        | (0,2) | (2,0) |
| 7        | (1,2) | (1,1) |
| 8        | (1,1) | (2,0) |
| ...      | ...   | ...   |
```

The table below gives the positions of the robot and the dust for the third testcase, where the expected output is 14.



```
| Timestep | Robot | Dust  |
| -------- | ----- | ----- |
| 0        | (0,0) | (1,1) |
| 1        | (0,1) | (2,2) |
| 2        | (0,2) | (1,1) |
| 3        | (0,3) | (3,0) |
| 4        | (1,3) | (1,1) |
| 5        | (2,3) | (2,2) |
| 6        | (3,3) | (1,1) |
| 7        | (3,2) | (3,0) |
| 8        | (3,1) | (1,1) |
| 9        | (3,0) | (2,2) |
| 10       | (2,0) | (1,1) |
| 11       | (2,1) | (3,0) |
| 12       | (2,2) | (1,1) |
| 13       | (1,2) | (2,2) |
| 14       | (1,1) | (1,1) |
```

Tourist Taxi Service



This problem is sponsored by Huawei.

Bob runs a taxi service that caters to tourists. He guarantees that all of his rides will begin with five unique streets in his historic city. Otherwise, his rides will be the shortest possible to a destination given this constraint. Your task is to find the length of time for all possible rides starting at a fixed location.

Standard input

Each input has a single test case.

The first line of input contains a single integer N, which gives the number of intersections in the city. Intersections are numbered 1 through N. All trips start at intersection 1.

The next line contains a single integer M which gives the number of streets in the city.

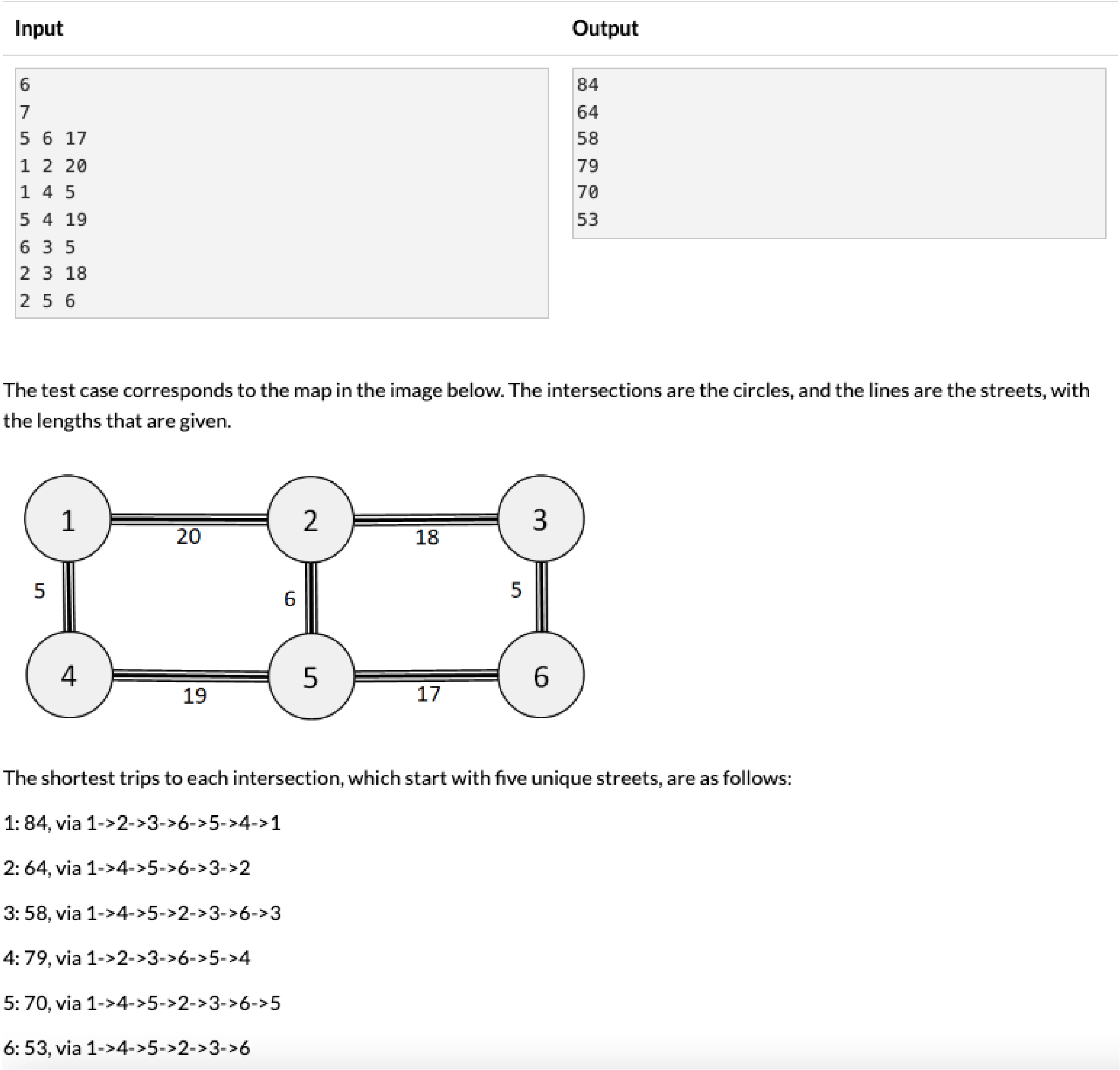
The following M lines contain three integers S, D, and L, and describe a two-way street that connects intersections S and D, with a time needed to traverse the street of L units.

Standard output

The output will contain N lines, each with a single integer. The i^{th} line will be the length of the shortest path from intersection 1 to intersection i, which begins with 5 unique streets.

Constraints and notes

- $4 \leq N \leq 10^4$
- $5 \leq M \leq 2 * 10^4$
- $1 \leq L_i \leq 10^5$
- $1 <= S_i, D_i <= N$
- Each intersection will be connected to at most 4 streets.
- It is guaranteed that any intersection can be reached from intersection 1.
- The street (S_i, D_i) is guaranteed to be unique in the input.
- It is guaranteed that there will always be paths that start with 5 distinct streets.



Input

```
6
7
5 6 17
1 2 20
1 4 5
5 4 19
6 3 5
2 3 18
2 5 6
```

Output

```
84
64
58
79
70
53
```

The test case corresponds to the map in the image below. The intersections are the circles, and the lines are the streets, with the lengths that are given.

The shortest trips to each intersection, which start with five unique streets, are as follows:

1: 84, via 1->2->3->6->5->4->1

2: 64, via 1->4->5->6->3->2

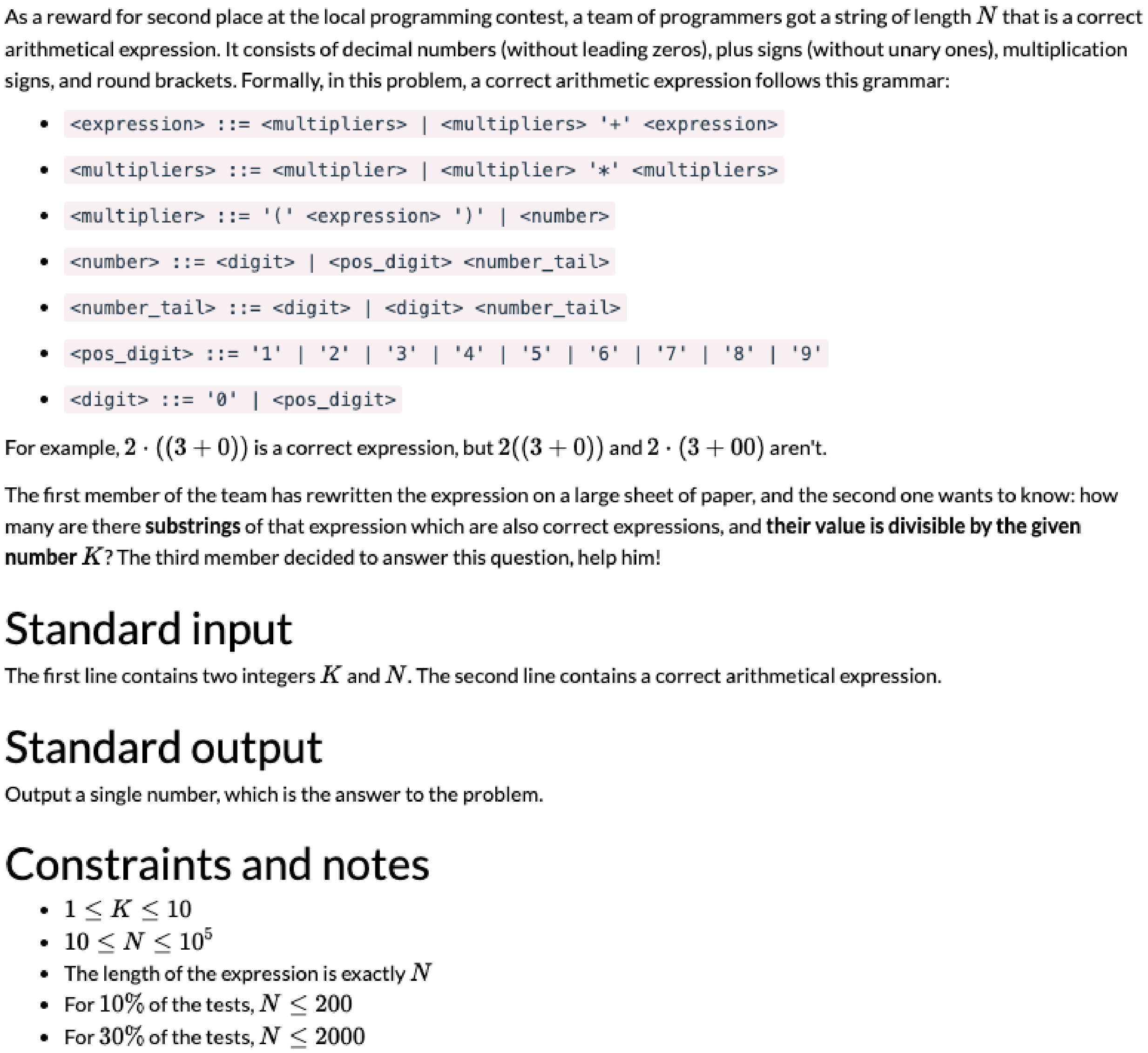
3: 58, via 1->4->5->2->3->6->3

4: 79, via 1->2->3->6->5->4

5: 70, via 1->4->5->2->3->6->5

6: 53, via 1->4->5->2->3->6

Weird Reward



As a reward for second place at the local programming contest, a team of programmers got a string of length N that is a correct arithmetical expression. It consists of decimal numbers (without leading zeros), plus signs (without unary ones), multiplication signs, and round brackets. Formally, in this problem, a correct arithmetic expression follows this grammar:

- `<expression> ::= <multipliers> | <multipliers> '+' <expression>`
- `<multipliers> ::= <multiplier> | <multiplier> '*' <multipliers>`
- `<multiplier> ::= '(' <expression> ')' | <number>`
- `<number> ::= <digit> | <pos_digit> <number_tail>`
- `<number_tail> ::= <digit> | <digit> <number_tail>`
- `<pos_digit> ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'`
- `<digit> ::= '0' | <pos_digit>`

For example, $2 \cdot ((3+0))$ is a correct expression, but $2((3+0))$ and $2 \cdot (3+00)$ aren't.

The first member of the team has rewritten the expression on a large sheet of paper, and the second one wants to know: how many are there **substrings** of that expression which are also correct expressions, and **their value is divisible by the given number K**? The third member decided to answer this question, help him!

Standard input

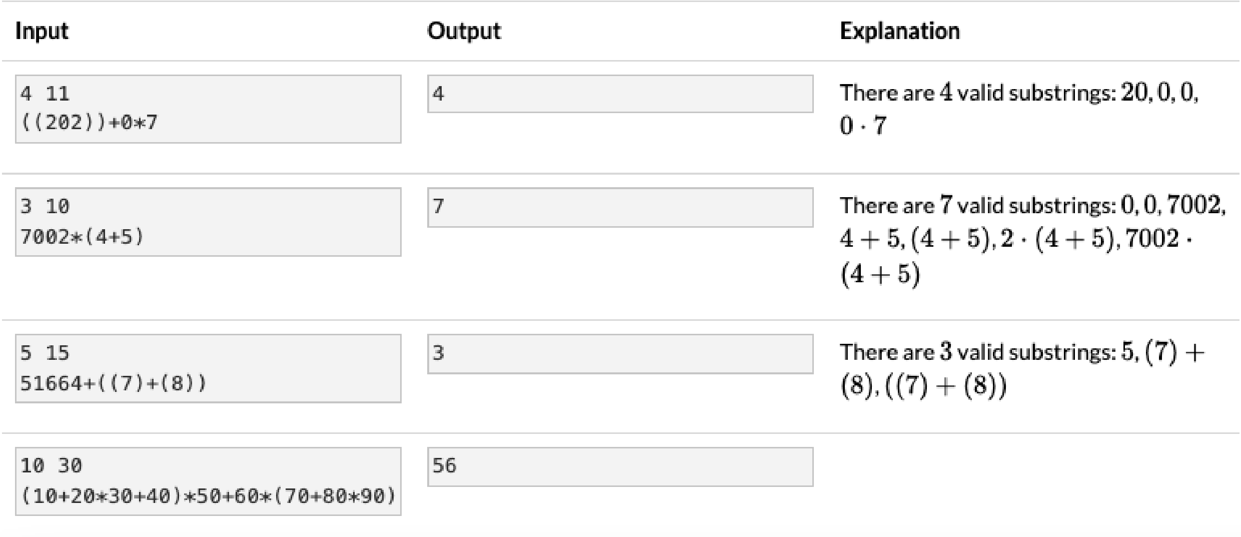
The first line contains two integers K and N. The second line contains a correct arithmetical expression.

Standard output

Output a single number, which is the answer to the problem.

Constraints and notes

- $1 \leq K \leq 10$
- $10 \leq N \leq 10^5$
- The length of the expression is exactly N
- For 10% of the tests, $N \leq 200$
- For 30% of the tests, $N \leq 2000$



Input	Output	Explanation

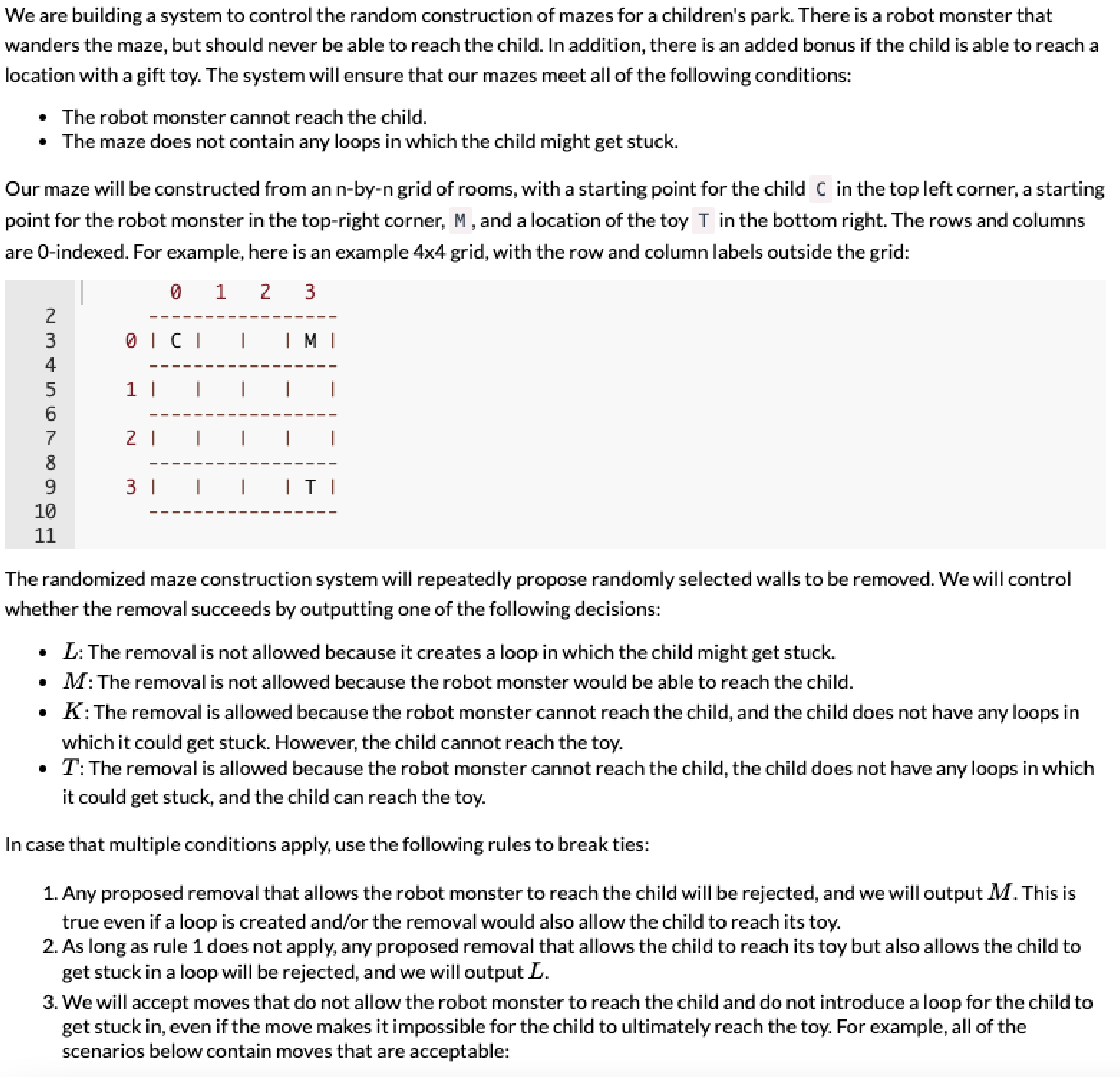
`4 11`
`((202))+0*7`
	`4`	There are 4 valid substrings: $20, 0, 0, 0 \cdot 7$

`3 10`
`7002*(4+5)`
	`7`	There are 7 valid substrings: $0, 0, 7002, 4+5, (4+5), 2 \cdot (4+5), 7002 \cdot (4+5)$

`5 15`
`51664+((7)+(8))`
	`3`	There are 3 valid substrings: $5, (7)+(8), ((7)+(8))$

`10 30`
`(10+20*30+40)*50+60*(70+80*90)`
	`56`	

Random Mazes



We are building a system to control the random construction of mazes for a children's park. There is a robot monster that wanders the maze, but should never be able to reach the child. In addition, there is an added bonus if the child is able to reach a location with a gift toy. The system will ensure that our mazes meet all of the following conditions:

- The robot monster cannot reach the child.
- The maze does not contain any loops in which the child might get stuck.

Our maze will be constructed from an n-by-n grid of rooms, with a starting point for the child `C` in the top left corner, a starting point for the robot monster in the top-right corner, `M`, and a location of the toy `T` in the bottom right. The rows and columns are 0-indexed. For example, here is an example 4x4 grid, with the row and column labels outside the grid:

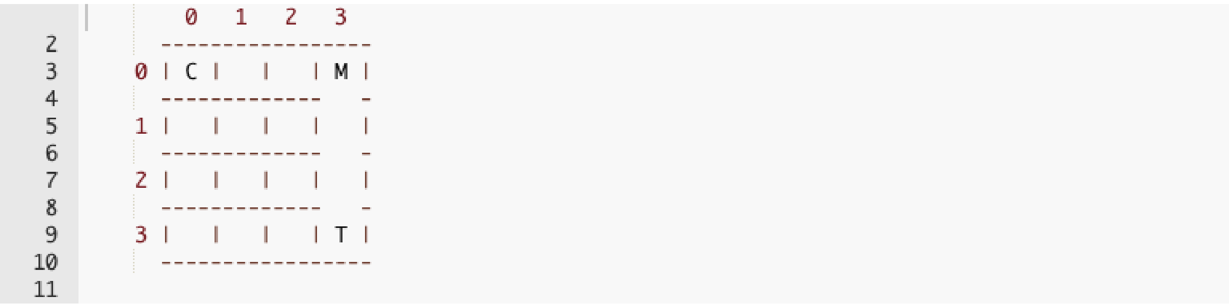
```
      0   1   2   3
    -----------------
  0 | C |   |   | M |
    -----------------
  1 |   |   |   |   |
    -----------------
  2 |   |   |   |   |
    -----------------
  3 |   |   |   | T |
    -----------------
```

The randomized maze construction system will repeatedly propose randomly selected walls to be removed. We will control whether the removal succeeds by outputting one of the following decisions:

- L: The removal is not allowed because it creates a loop in which the child might get stuck.
- M: The removal is not allowed because the robot monster would be able to reach the child.
- K: The removal is allowed because the robot monster cannot reach the child, and the child does not have any loops in which it could get stuck. However, the child cannot reach the toy.
- T: The removal is allowed because the robot monster cannot reach the child, the child does not have any loops in which it could get stuck, and the child can reach the toy.

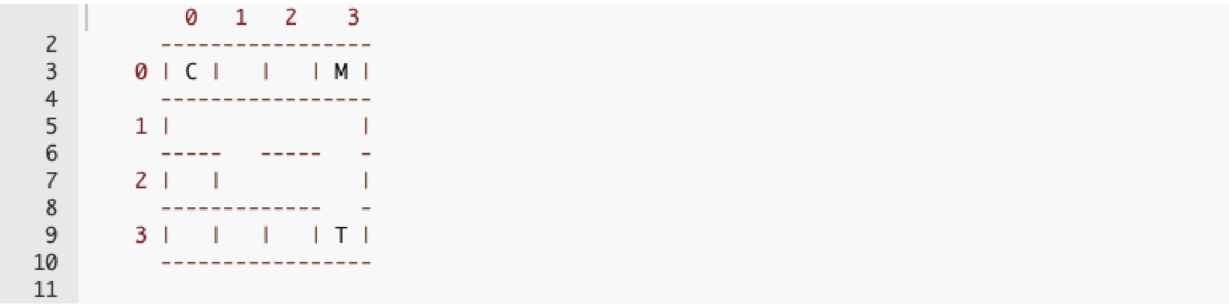
In case that multiple conditions apply, use the following rules to break ties:

1. Any proposed removal that allows the robot monster to reach the child will be rejected, and we will output M. This is true even if a loop is created and/or the removal would also allow the child to reach its toy.
2. As long as rule 1 does not apply, any proposed removal that allows the child to reach its toy but also allows the child to get stuck in a loop will be rejected, and we will output L.
3. We will accept moves that do not allow the robot monster to reach the child and do not introduce a loop for the child to get stuck in, even if the move makes it impossible for the child to ultimately reach the toy. For example, all of the scenarios below contain moves that are acceptable:



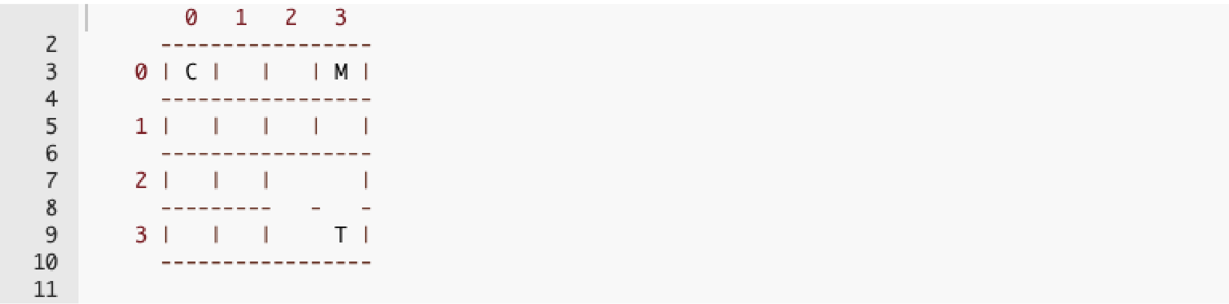
```
    0   1   2   3
  -----------------
0 | C |   |   | M |
  -------------   -
1 |   |   |   |   |
  -------------   -
2 |   |   |   |   |
  -------------   -
3 |   |   |   | T |
  -----------------
```

The maze above is acceptable, even though there is no way for the child to safely reach the toy.



```
    0   1   2    3
  -----------------
0 | C |   |   | M |
  -----------------
1 |               |
  -----   -----   -
2 |   |           |
  -------------   -
3 |   |   |   | T |
  -----------------
```

The maze above is acceptable, even though there is no way for the child to reach the toy, since it would have to pass through a region with a loop.



```
    0   1   2   3
  -----------------
0 | C |   |   | M |
  -----------------
1 |   |   |   |   |
  -----------------
2 |   |   |       |
  ---------   -   -
3 |   |   |     T |
  -----------------
```

Similarly, the maze above is acceptable, even though there is no way for the child to reach the toy, since the toy is located in a region with a loop.

Standard input

Each input has a single test case.

The input begins with a line containing two space-separated integers, n and w. n specifies the length and width of the maze, and w specifies how many wall removals the system will propose.

Then come w lines, each representing a proposed wall to remove, in the following format: `r c DIR`

This proposal represents a proposal to remove a wall from the room at row r and column c. DIR will be chosen from the set { N,E,W,S}, indicating which wall should be removed. N,E,W, and S correspond to the north, east, west, or south wall (respectively). North corresponds to the top of the maze, East to the right of the maze, and so on.

Standard output

The output will contain w lines, containing the w responses to the maze creation system's proposals. Each line will contain the appropriate symbol chosen from the set: { `L` , `M` , `K` , `T` }.

Constraints and notes

- $2 \leq N \leq 500$
- $1 \leq w \leq 400,000$
- $0 \leq r, c \leq N - 1$

The maze construction system will never propose that an outer wall should be removed. For example, it would not propose `0 0 N`

The system will never propose that the same wall should be removed twice. It would not, for example, ask to remove the east wall of room (0,0) twice. Similarly, it would not ask to remove the east wall of room (1,1), and then ask to remove the west wall of room (1,2), since this is the same wall.

Input

```
4 19
1 3 S
2 2 N
2 2 E
1 2 E
0 0 E
0 1 E
0 2 E
3 0 E
0 2 S
3 3 W
0 3 S
3 2 W
1 0 N
1 0 S
3 0 N
1 1 W
1 1 E
1 1 S
3 1 N
```

Output

```
K
K
K
K
K
K
M
K
L
K
K
K
K
K
T
T
M
T
L
```

Proposal 1: `1 3 S`

After breaking down the South wall at (1,3), the maze looks like:

```
    0   1   2   3
  -----------------
0 | C |   |   | M |
  -----------------
1 |   |   |   |   |
  -------------   -
2 |   |   |   |   |
  -----------------
3 |   |   |   | T |
  -----------------

```

No constraints are violated, so we output: K

Proposal 2: `2 2 N`

After breaking down the North wall at (2,2), the maze looks like:

```
        0   1   2   3
      -----------------
    0 | C |   |   | M |
      -----------------
    1 |   |   |   |   |
      ---------   -   -
    2 |   |   |   |   |
      -----------------
    3 |   |   |   | T |
      -----------------

```

No constraints are violated, so we output: K

Proposal 3: `2 2 E`

After breaking down the East wall at (2,2), the maze looks like:

```
        0   1   2   3
      -----------------
    0 | C |   |   | M |
      -----------------
    1 |   |   |   |   |
      ---------   -   -
    2 |   |   |       |
      -----------------
    3 |   |   |   | T |
      -----------------

```

No constraints are violated, so we output: K

Proposal 4: `1 2 E`

After breaking down the East wall at (1,2), the maze looks like:

```
        0   1   2   3
      -----------------
    0 | C |   |   | M |
      -----------------
    1 |   |   |       |
      ---------   -   -
    2 |   |   |       |
      -----------------
    3 |   |   |   | T |
      -----------------

```

Note that there is a loop in the maze, but the child cannot reach the loop, so it is ok. No constraints are violated, so we output: K

Proposal 5: `0 0 E`

After breaking down the East wall at (0,0), the maze looks like:

```
        0   1   2    3
      -----------------
    0 | C     |   | M |
      -----------------
    1 |   |   |       |
      ---------   -   -
    2 |   |   |       |
      -----------------
    3 |   |   |   | T |
      -----------------

```

No constraints are violated, so we output: K

Proposal 6: `0 1 E`

After breaking down the East wall at (0,1), the maze looks like:

```
    0   1   2   3
  -----------------
0 | C         | M |
  -----------------
1 |   |   |       |
  ---------   -   -
2 |   |   |       |
  -----------------
3 |   |   |   | T |
  -----------------

```

No constraints are violated, so we output: K

Proposal 7: `0 2 E`

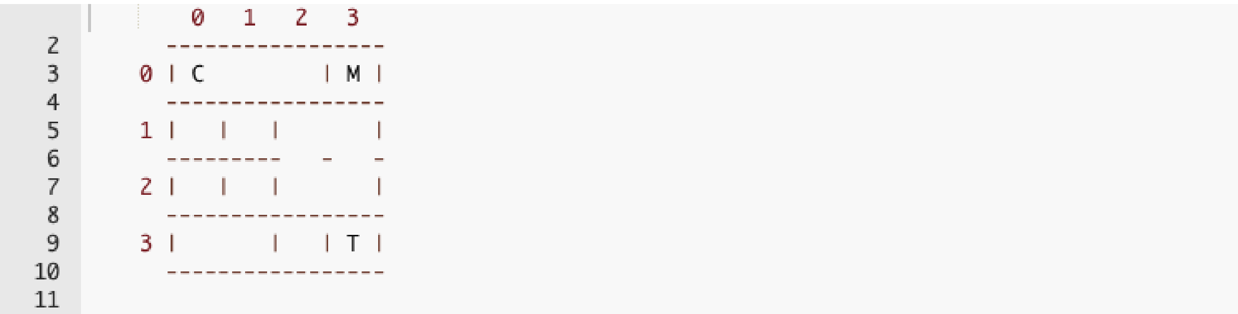
If we were to break down the East wall at (0,2), the maze would look like:

```
    0   1   2   3
  -----------------
0 | C           M |
  -----------------
1 |   |   |       |
  ---------   -   -
2 |   |   |       |
  -----------------
3 |   |   |   | T |
  -----------------

```

Note that the robot monster would be able to get to the child, so we reject this proposal. The maze looks exactly like it did after **Proposal 6** and we output: M

Proposal 8: `3 0 E`

After breaking down the East wall at (3,0), the maze looks like:

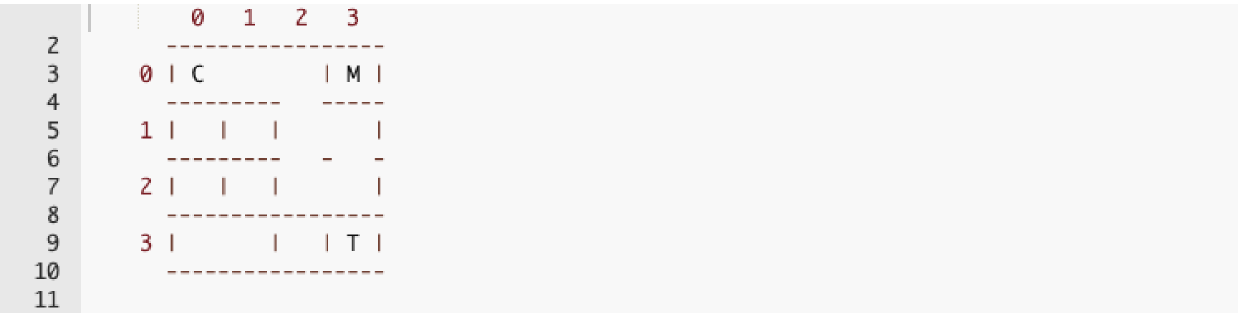


```
        0   1   2   3
      -----------------
    0 | C         | M |
      -----------------
    1 |   |   |       |
      ---------   -   -
    2 |   |   |       |
      -----------------
    3 |       |   | T |
      -----------------
```

No constraints are violated, so we output: K

Proposal 9: `0 2 S`

If we were to break down the South wall at (0,2), the maze would look like:



```
        0   1   2   3
      -----------------
    0 | C         | M |
      ---------   -----
    1 |   |   |       |
      ---------   -   -
    2 |   |   |       |
      -----------------
    3 |       |   | T |
      -----------------
```

Note that the child could get caught in a loop here, so we do not allow this change. The maze looks exactly like it did after **Proposal 8** and we output: L

Proposal 10: `3 3 W`

After breaking down the West wall at (3,3), the maze looks like:

```
    0   1   2   3
  -----------------
0 | C         | M |
  -----------------
1 |   |   |       |
  ---------   -   -
2 |   |   |       |
  -----------------
3 |       |     T |
  -----------------

```

No constraints are violated, so we output: K

Proposal 11: `0 3 S`

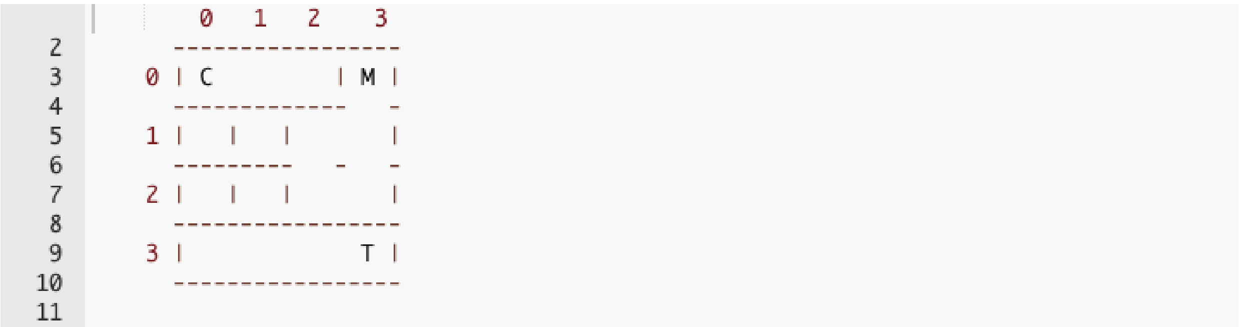
After breaking down the South wall at (0,3), the maze looks like:

```
    0   1   2    3
  -----------------
0 | C         | M |
  -------------   -
1 |   |   |       |
  ---------   -   -
2 |   |   |       |
  -----------------
3 |       |     T |
  -----------------

```

Note that it is ok to have a loop in the maze that the robot monster might get stuck in. No constraints are violated, so we output: K

Proposal 12: `3 2 W`

After breaking down the West wall at (3,2), the maze looks like:

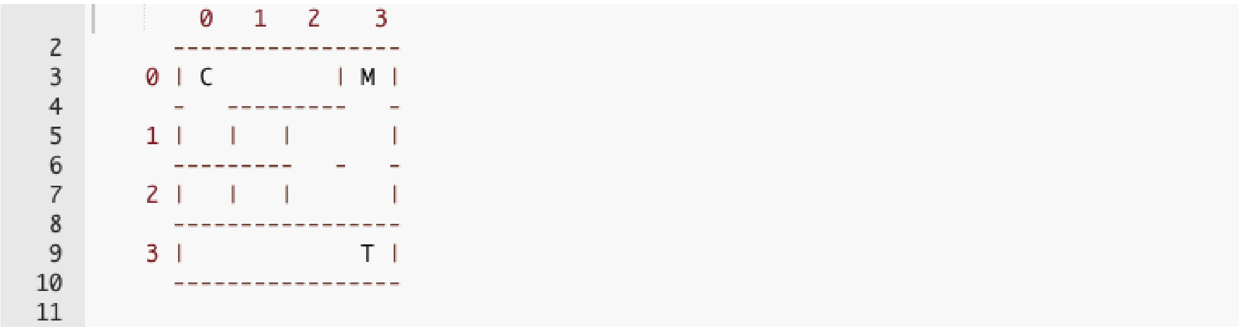


```
    0   1   2    3
  -----------------
0 | C         | M |
  -------------   -
1 |   |   |       |
  ---------   -   -
2 |   |   |       |
  -----------------
3 |             T |
  -----------------
```

No constraints are violated, so we output: K

Proposal 13: `1 0 N`

After breaking down the North wall at (1,0), the maze looks like:

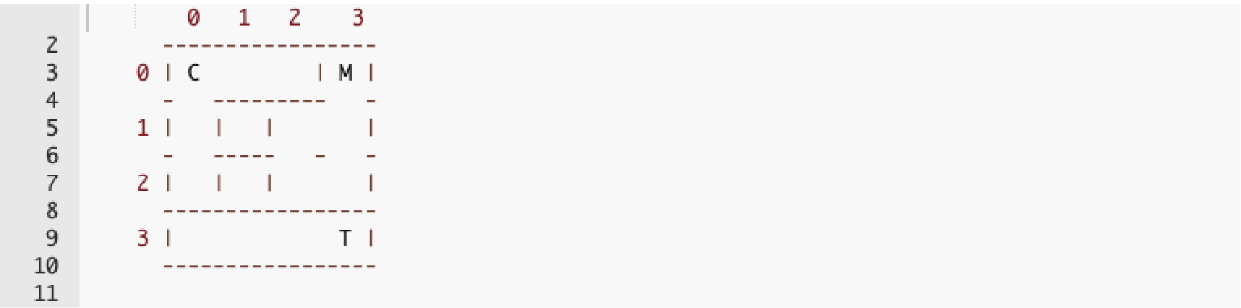


```
    0   1   2    3
  -----------------
0 | C         | M |
  -   ---------   -
1 |   |   |       |
  ---------   -   -
2 |   |   |       |
  -----------------
3 |             T |
  -----------------
```

No constraints are violated, so we output: K

Proposal 14: `1 0 S`

After breaking down the South wall at (1,0), the maze looks like:

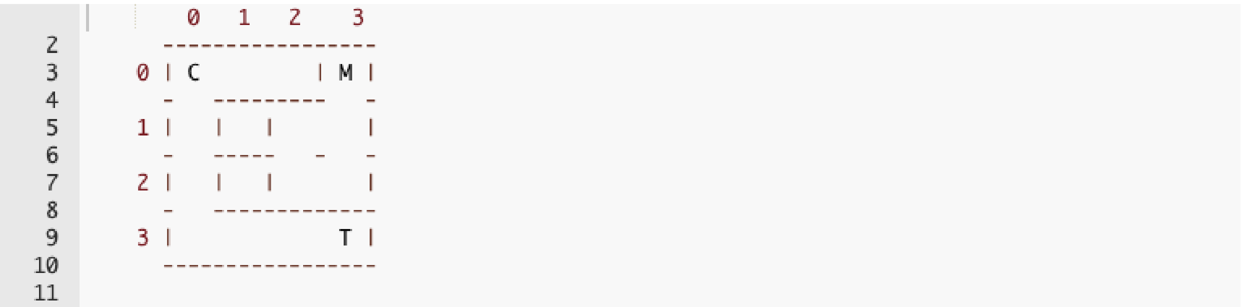


```
    0   1   2    3
  -----------------
0 | C         | M |
  -   ---------   -
1 |   |   |       |
  -   -----   -   -
2 |   |   |       |
  -----------------
3 |             T |
  -----------------
```

No constraints are violated, so we output: K

Proposal 15: `3 0 N`

After breaking down the North wall at (3,0), the maze looks like:

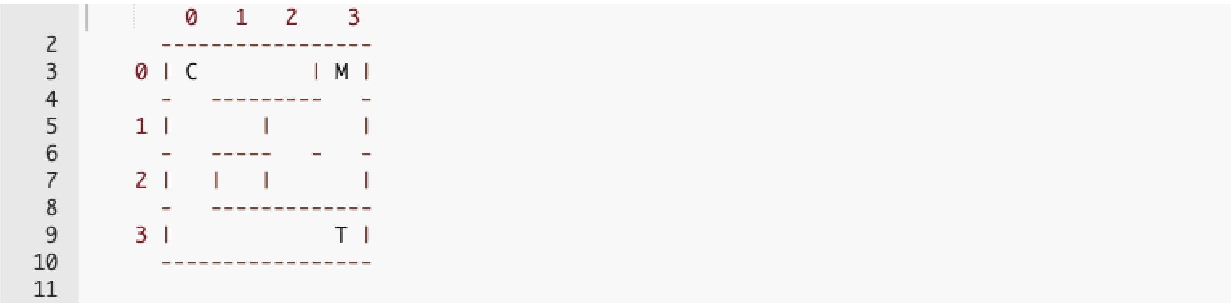


```
    0   1   2    3
  -----------------
0 | C         | M |
  -   ---------   -
1 |   |   |       |
  -   -----   -   -
2 |   |   |       |
  -   -------------
3 |             T |
  -----------------
```

No constraints are violated _and_ the child can reach the toy, so we output: T

Proposal 16: `1 1 W`

After breaking down the West wall at (1,1), the maze looks like:

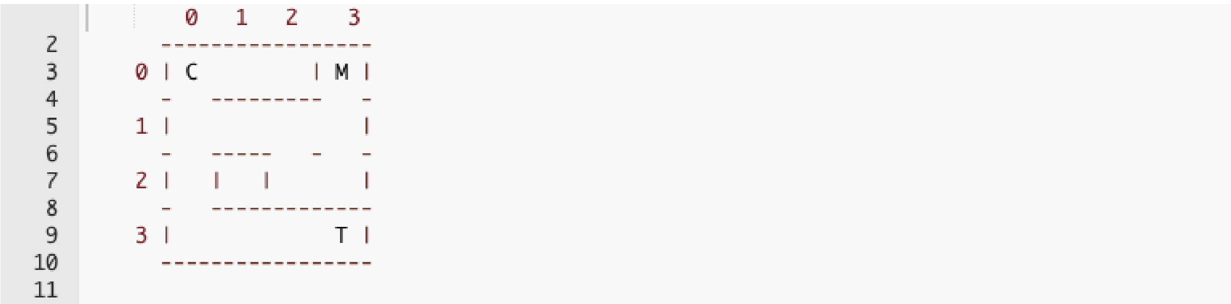


```
        0   1   2    3
      -----------------
    0 | C         | M |
      -   ---------   -
    1 |       |       |
      -   -----   -   -
    2 |   |   |       |
      -   -------------
    3 |             T |
      -----------------
```

No constraints are violated _and_ the child can reach the toy, so we output: T

Proposal 17: `1 1 E`

If we were to break down the east wall at (1,1), the maze would look like:

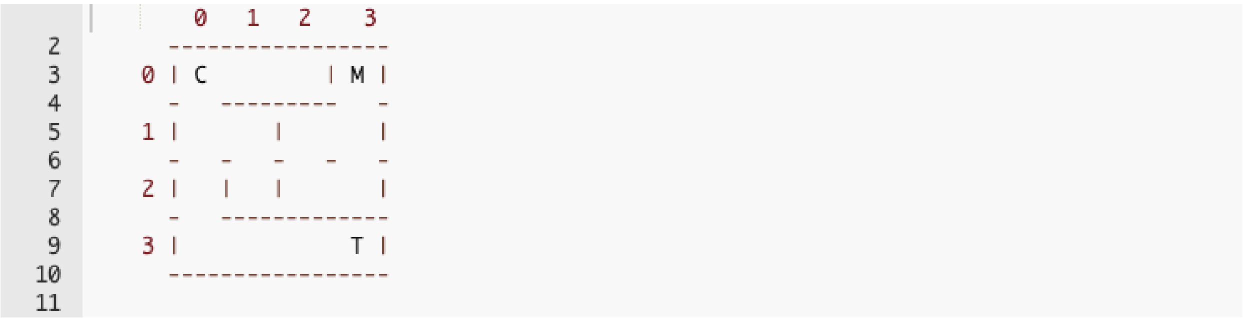


```
        0   1   2    3
      -----------------
    0 | C         | M |
      -   ---------   -
    1 |               |
      -   -----   -   -
    2 |   |   |       |
      -   -------------
    3 |             T |
      -----------------
```

The child could get to the toy, but they could get stuck in a loop and the robot monster could reach the child. In this case, we reject the proposal. The maze will be exactly as it was after **Proposal 16**. We output (see notes in the Introduction about breaking ties like this): M

Proposal 18: `1 1 S`

After breaking down the South wall at (1,1), the maze looks like:

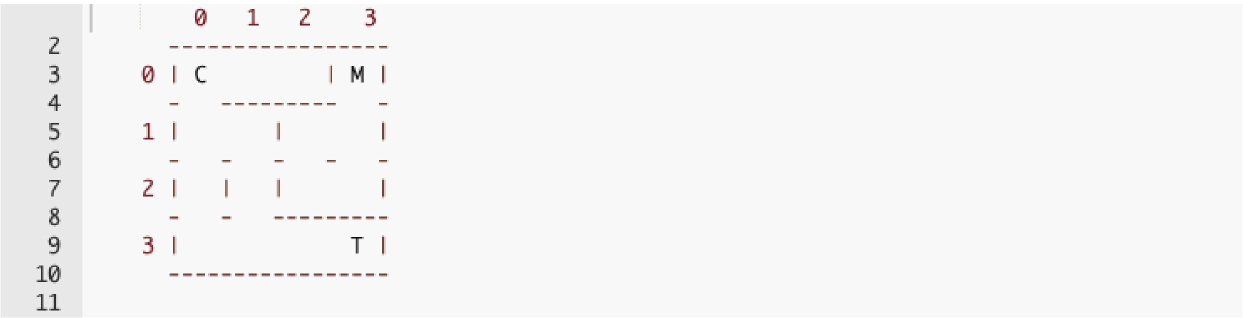


```
    0   1   2    3
  -----------------
0 | C         | M |
  -   ---------   -
1 |       |       |
  -   -   -   -   -
2 |   |   |       |
  -   -------------
3 |             T |
  -----------------
```

No constraints are violated _and_ the child can reach the toy, so we output: T

Proposal 19: `3 1 N`

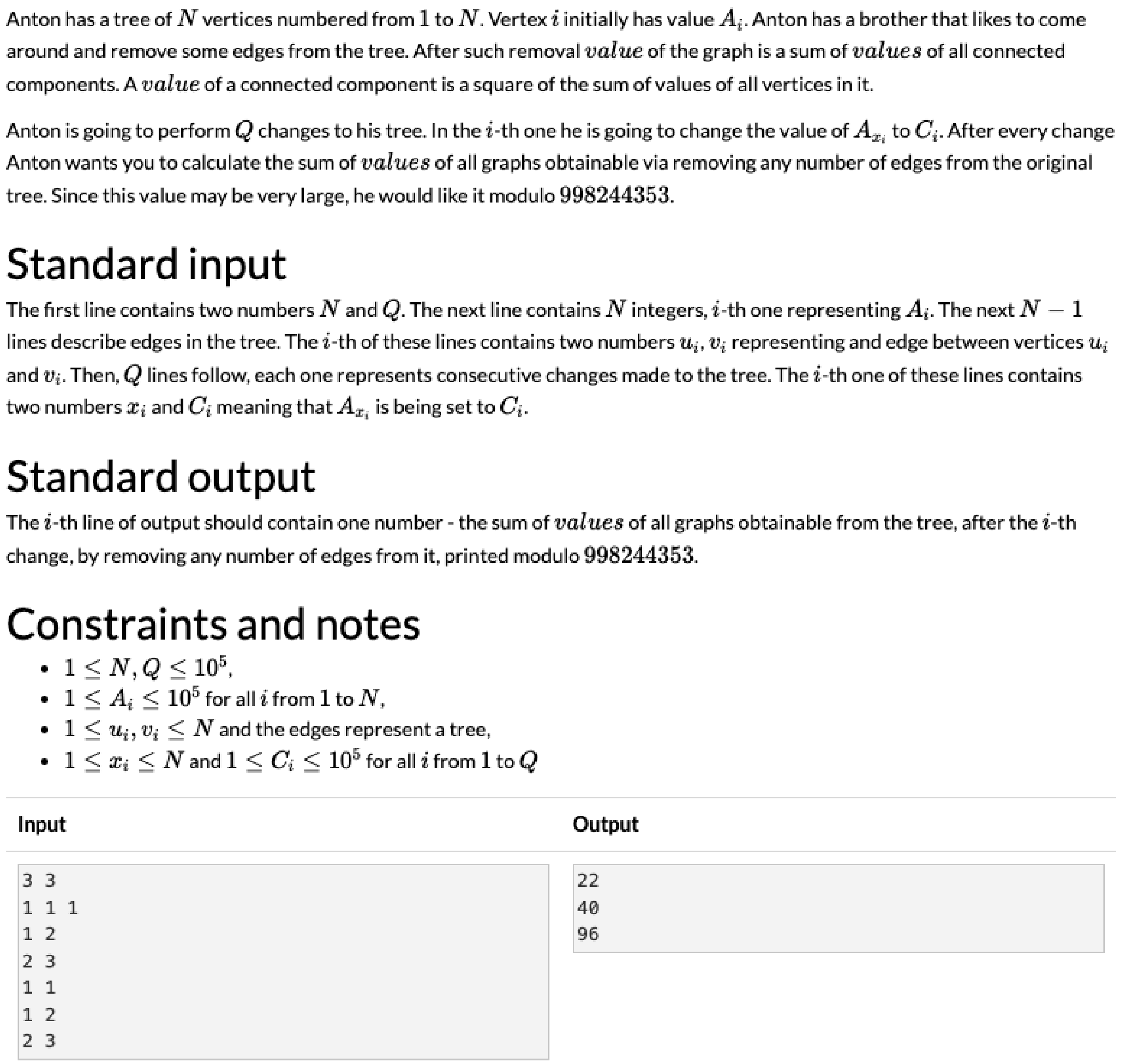
If we were to break down the North wall at (3,1). The maze would look like:



```
    0   1   2    3
  -----------------
0 | C         | M |
  -   ---------   -
1 |       |       |
  -   -   -   -   -
2 |   |   |       |
  -   -   ---------
3 |             T |
  -----------------
```

While the child could reach the toy, they could also get stuck in a loop. Thus, we reject this proposal, and output: L

Queries on a Tree



Anton has a tree of N vertices numbered from 1 to N. Vertex i initially has value A_i. Anton has a brother that likes to come around and remove some edges from the tree. After such removal $value$ of the graph is a sum of $values$ of all connected components. A $value$ of a connected component is a square of the sum of values of all vertices in it.

Anton is going to perform Q changes to his tree. In the i-th one he is going to change the value of A_{x_i} to C_i. After every change Anton wants you to calculate the sum of $values$ of all graphs obtainable via removing any number of edges from the original tree. Since this value may be very large, he would like it modulo 998244353.

Standard input

The first line contains two numbers N and Q. The next line contains N integers, i-th one representing A_i. The next $N-1$ lines describe edges in the tree. The i-th of these lines contains two numbers u_i, v_i representing and edge between vertices u_i and v_i. Then, Q lines follow, each one represents consecutive changes made to the tree. The i-th one of these lines contains two numbers x_i and C_i meaning that A_{x_i} is being set to C_i.

Standard output

The i-th line of output should contain one number - the sum of $values$ of all graphs obtainable from the tree, after the i-th change, by removing any number of edges from it, printed modulo 998244353.

Constraints and notes

- $1 \leq N, Q \leq 10^5$,
- $1 \leq A_i \leq 10^5$ for all i from 1 to N,
- $1 \leq u_i, v_i \leq N$ and the edges represent a tree,
- $1 \leq x_i \leq N$ and $1 \leq C_i \leq 10^5$ for all i from 1 to Q

Input

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

Output

```
22
40
96
```