Good Bye 2016

A. New Year and Hurry

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Limak is going to participate in a contest on the last day of the 2016. The contest will start at 20:00 and will last four hours, exactly until midnight. There will be n problems, sorted by difficulty, i.e. problem 1 is the easiest and problem n is the hardest. Limak knows it will take him $5 \cdot i$ minutes to solve the i-th problem.

Limak's friends organize a New Year's Eve party and Limak wants to be there at midnight or earlier. He needs k minutes to get there from his house, where he will participate in the contest first.

How many problems can Limak solve if he wants to make it to the party?

Input

The only line of the input contains two integers n and k $(1 \le n \le 10, 1 \le k \le 240)$ — the number of the problems in the contest and the number of minutes Limak needs to get to the party from his house.

Output

Print one integer, denoting the maximum possible number of problems Limak can solve so that he could get to the party at midnight or earlier.

Examples

input
```
3 222
```
output
```
2
```

input
```
4 190
```
output
```
4
```

input
```
7 1
```
output
```
7
```

Note

In the first sample, there are 3 problems and Limak needs 222 minutes to get to the party. The three problems require 5, 10 and 15 minutes respectively. Limak can spend $5 + 10 = 15$ minutes to solve first two problems. Then, at 20:15 he can leave his house to get to the party at 23:57 (after 222 minutes). In this scenario Limak would solve 2 problems. He doesn't have enough time to solve 3 problems so the answer is 2.

In the second sample, Limak can solve all 4 problems in $5 + 10 + 15 + 20 = 50$ minutes. At 20:50 he will leave the house and go to the party. He will get there exactly at midnight.

In the third sample, Limak needs only 1 minute to get to the party. He has enough time to solve all 7 problems.

B. New Year and North Pole

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

In this problem we assume the Earth to be a completely round ball and its surface a perfect sphere. The length of the equator and any meridian is considered to be exactly $40\,000$ kilometers. Thus, travelling from North Pole to South Pole or vice versa takes exactly $20\,000$ kilometers.

Limak, a polar bear, lives on the North Pole. Close to the New Year, he helps somebody with delivering packages all around the world. Instead of coordinates of places to visit, Limak got a description how he should move, assuming that he starts from the North Pole. The description consists of n parts. In the i-th part of his journey, Limak should move t_i kilometers in the direction represented by a string dir_i that is one of: "`North`", "`South`", "`West`", "`East`".

Limak isn't sure whether the description is valid. You must help him to check the following conditions:

- If at any moment of time (before any of the instructions or while performing one of them) Limak is on the North Pole, he can move only to the South.
- If at any moment of time (before any of the instructions or while performing one of them) Limak is on the South Pole, he can move only to the North.
- The journey must end on the North Pole.

Check if the above conditions are satisfied and print "`YES`" or "`NO`" on a single line.

Input

The first line of the input contains a single integer n $(1 \le n \le 50)$.

The i-th of next n lines contains an integer t_i and a string dir_i $(1 \le t_i \le 10^6,)$ — the length and the direction of the i-th part of the journey, according to the description Limak got.

Output

Print "`YES`" if the description satisfies the three conditions, otherwise print "`NO`", both without the quotes.

Examples

input

```
5
7500 South
10000 East
3500 North
4444 West
4000 North
```

output

```
YES
```

input

```
2
15000 South
4000 East
```

output

```
NO
```

input

```
5
20000 South
1000 North
1000000 West
9000 North
10000 North
```

output

```
YES
```

input

```
3
20000 South
10 East
20000 North
```

output

```
NO
```

input

```
2
```

```
1000 North
1000 South
```

output

```
NO
```

input

```
4
50 South
50 North
15000 South
15000 North
```

output

```
YES
```

Note

Drawings below show how Limak's journey would look like in first two samples. In the second sample the answer is "NO" because he doesn't end on the North Pole.

C. New Year and Rating

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Every Codeforces user has rating, described with one integer, possibly negative or zero. Users are divided into two divisions. The first division is for users with rating 1900 or higher. Those with rating 1899 or lower belong to the second division. In every contest, according to one's performance, his or her rating changes by some value, possibly negative or zero.

Limak competed in n contests in the year 2016. He remembers that in the i-th contest he competed in the division d_i (i.e. he belonged to this division **just before** the start of this contest) and his rating changed by c_i **just after the contest**. Note that negative c_i denotes the loss of rating.

What is the maximum possible rating Limak can have right now, after all n contests? If his rating may be arbitrarily big, print "`Infinity`". If there is no scenario matching the given information, print "`Impossible`".

Input

The first line of the input contains a single integer n ($1 \le n \le 200\,000$).

The i-th of next n lines contains two integers c_i and d_i ($-100 \le c_i \le 100$, $1 \le d_i \le 2$), describing Limak's rating change after the i-th contest and his division during the i-th contest contest.

Output

If Limak's current rating can be arbitrarily big, print "`Infinity`" (without quotes). If the situation is impossible, print "`Impossible`" (without quotes). Otherwise print one integer, denoting the maximum possible value of Limak's current rating, i.e. rating after the n contests.

Examples

input
```
3
-7 1
5 2
8 2
```
output
```
1907
```

input
```
2
57 1
22 2
```
output
```
Impossible
```

input
```
1
-5 1
```
output
```
Infinity
```

input
```
4
27 2
13 1
-50 1
8 2
```
output
```
1897
```

Note

In the first sample, the following scenario matches all information Limak remembers and has maximum possible final rating:

- Limak has rating 1901 and belongs to the division 1 in the first contest. His rating decreases by 7.
- With rating 1894 Limak is in the division 2. His rating increases by 5.
- Limak has rating 1899 and is still in the division 2. In the last contest of the year he gets $+8$ and ends the year with rating 1907.

In the second sample, it's impossible that Limak is in the division 1, his rating increases by 57 and after that Limak is in the division 2 in the second contest.

D. New Year and Fireworks

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

One tradition of welcoming the New Year is launching fireworks into the sky. Usually a launched firework flies vertically upward for some period of time, then explodes, splitting into several parts flying in different directions. Sometimes those parts also explode after some period of time, splitting into even more parts, and so on.

Limak, who lives in an infinite grid, has a single firework. The behaviour of the firework is described with a recursion depth n and a duration for each level of recursion $t_1, t_2, ..., t_n$. Once Limak launches the firework in some cell, the firework starts moving upward. After covering t_1 cells (including the starting cell), it explodes and splits into two parts, each moving in the direction changed by 45 degrees (see the pictures below for clarification). So, one part moves in the top-left direction, while the other one moves in the top-right direction. Each part explodes again after covering t_2 cells, splitting into two parts moving in directions again changed by 45 degrees. The process continues till the n-th level of recursion, when all 2^{n-1} existing parts explode and disappear without creating new parts. After a few levels of recursion, it's possible that some parts will be at the same place and at the same time — it is allowed and such parts do not crash.

Before launching the firework, Limak must make sure that nobody stands in cells which will be visited at least once by the firework. Can you count the number of those cells?

Input

The first line of the input contains a single integer n $(1 \le n \le 30)$ — the total depth of the recursion.

The second line contains n integers $t_1, t_2, ..., t_n$ $(1 \le t_i \le 5)$. On the i-th level each of 2^{i-1} parts will cover t_i cells before exploding.

Output

Print one integer, denoting the number of cells which will be visited at least once by any part of the firework.

Examples

input
```
4
4 2 2 3
```
output
```
39
```

input
```
6
1 1 1 1 1 3
```
output
```
85
```

input
```
1
3
```
output
```
3
```

Note

For the first sample, the drawings below show the situation after each level of recursion. Limak launched the firework from the bottom-most red cell. It covered $t_1 = 4$ cells (marked red), exploded and divided into two parts (their further movement is marked green). All explosions are marked with an 'X' character. On the last drawing, there are 4 red, 4 green, 8 orange and 23 pink cells. So, the total number of visited cells is $4 + 4 + 8 + 23 = 39$.

For the second sample, the drawings below show the situation after levels 4, 5 and 6. The middle drawing shows directions of all parts that will move in the next level.

E. New Year and Old Subsequence

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A string t is called *nice* if a string "2017" occurs in t as a **subsequence** but a string "2016" doesn't occur in t as a **subsequence**. For example, strings "203434107" and "9220617" are nice, while strings "20016", "1234" and "20167" aren't nice.

The *ugliness* of a string is the minimum possible number of characters to remove, in order to obtain a nice string. If it's impossible to make a string nice by removing characters, its ugliness is -1.

Limak has a string s of length n, with characters indexed 1 through n. He asks you q queries. In the i-th query you should compute and print the ugliness of a **substring** (continuous subsequence) of s starting at the index a_i and ending at the index b_i (inclusive).

Input

The first line of the input contains two integers n and q ($4 \le n \le 200\,000$, $1 \le q \le 200\,000$) — the length of the string s and the number of queries respectively.

The second line contains a string s of length n. Every character is one of digits '0'–'9'.

The i-th of next q lines contains two integers a_i and b_i ($1 \le a_i \le b_i \le n$), describing a substring in the i-th query.

Output

For each query print the ugliness of the given substring.

Examples

input

```
8 3
20166766
1 8
1 7
2 8
```

output

```
4
3
-1
```

input

```
15 5
012016662091670
3 4
1 14
4 15
1 13
10 15
```

output

```
-1
2
1
-1
-1
```

input

```
4 2
1234
2 4
1 2
```

output

```
-1
-1
```

Note

In the first sample:

- In the first query, $ugliness($"20166766"$) = 4$ because all four sixes must be removed.
- In the second query, $ugliness($"2016676"$) = 3$ because all three sixes must be removed.
- In the third query, $ugliness($"0166766"$) = -1$ because it's impossible to remove some digits to get a nice string.

In the second sample:

- In the second query, $ugliness($"01201666209167"$) = 2$. It's optimal to remove the first digit '2' and the last digit '6', what gives a string "010166620917", which is nice.

- In the third query, $ugliness($"`016662091670`"$) = 1$. It's optimal to remove the last digit '`6`', what gives a nice string "`01666209170`".

F. New Year and Finding Roots

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem. In the interaction section below you will find the information about flushing the output.

The New Year tree of height h is a perfect binary tree with vertices numbered 1 through $2^h - 1$ in some order. In this problem we assume that h is at least 2. The drawing below shows one example New Year tree of height 3:

Polar bears love decorating the New Year tree and Limak is no exception. To decorate the tree, he must first find its root, i.e. a vertex with exactly two neighbours (assuming that $h \geq 2$). It won't be easy because Limak is a little bear and he doesn't even see the whole tree. Can you help him?

There are t testcases. In each testcase, you should first read h from the input. Then you can ask at most 16 questions of format "`? x`" (without quotes), where x is an integer between 1 and $2^h - 1$, inclusive. As a reply you will get the list of neighbours of vertex x (more details in the "Interaction" section below). For example, for a tree on the drawing above after asking "`? 1`" you would get a response with 3 neighbours: 4, 5 and 7. Your goal is to find the index of the root y and print it in the format "`! y`". You will be able to read h for a next testcase only after printing the answer in a previous testcase and flushing the output.

Each tree is fixed from the beginning and it doesn't change during your questions.

Input

The first line of the input contains a single integer t $(1 \leq t \leq 500)$ — the number of testcases.

At the beginning of each testcase you should read from the input a single integer h $(2 \leq h \leq 7)$ — the height of the tree. You can't read the value of h in a next testcase until you answer a previous testcase.

Interaction

To ask a question about neighbours of vertex x, print "`? x`" (without quotes) on a separate line. Note, you must print an end-of-line character after the last character of the line and flush your output to get a response.

The response will consist of two lines. The first line will contain a single integer k $(1 \leq k \leq 3)$ — the number of neighbours of vertex x. The second line will contain k distinct integers $t_1, ..., t_k$ $(1 \leq t_1 < ... < t_k \leq 2^h - 1)$ — indices of neighbours of vertex x, gives in the increasing order.

After asking at most 16 questions you have to say y — the index of the root. Print "`! y`" (without quotes) and an end-of-line character, and flush the output.

Each tree is fixed from the beginning and it doesn't change during your questions.

You can get `Idleness Limit Exceeded` if you don't print anything or if you forget to flush the output.

To flush you can use (just printing a query/answer and end-of-line):

- `fflush(stdout)` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python;
- `flush(output)` in Pascal;
- See the documentation for other languages.

In any moment if the program reads $h = 0$ or $k = 0$ it should immediately terminate normally (for example, calling exit(0)). It means that the system detected incorrect request/output from your program and printed 0 because if can't process your requests anymore. In this case you'll receive verdict "Wrong Answer", but if you ignore case $h = 0$ or $k = 0$ it could lead to "Runtime Error", "Time/Memory limit exceeded" or any other verdict because your program could read a trash from the closed input stream.

Hacking. To hack someone, use the following format:

The first line should contain a single integer t equal to 1 (only one testcase is allowed in hacks). The second line should contain a single integer h. Each of next $2^h - 2$ lines should contain two distinct integers a_i and b_i $(1 \leq a_i, b_i \leq 2^h - 1)$, denoting two nodes connected with an edge. The printed edges must form a perfect binary tree of height h.

Of course, contestant programs will not be able to see this input.

Examples

input

```
1
3
3
4 5 7
2
1 2
1
2
```

output

```
? 1
? 5
? 6
! 5
```

input

```
2
2
1
3
2
1 2
2
1 2
4
3
3 12 13
```

output

```
? 1
? 3
? 3
! 3
? 6
! 1
```

Note

In the first sample, a tree corresponds to the drawing from the statement.

In the second sample, there are two two testcases. A tree in the first testcase has height 2 and thus 3 vertices. A tree in the second testcase has height 4 and thus 15 vertices. You can see both trees on the drawing below.

G. New Year and Binary Tree Paths

time limit per test: 3 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

The New Year tree is an infinite perfect binary tree rooted in the node 1. Each node v has two children: nodes indexed $(2 \cdot v)$ and $(2 \cdot v + 1)$.

Polar bears love decorating the New Year tree and Limak is no exception. As he is only a little bear, he was told to decorate only one simple path between some pair of nodes. Though he was given an opportunity to pick the pair himself! Now he wants to know the number of unordered pairs of indices (u, v) $(u \leq v)$, such that the sum of indices of all nodes along the simple path between u and v (including endpoints) is equal to s. Can you help him and count this value?

Input

The only line of the input contains a single integer s $(1 \leq s \leq 10^{15})$.

Output

Print one integer, denoting the number of unordered pairs of nodes indices defining simple paths with the sum of indices of vertices equal to s.

Example

input

```
10
```

output

```
4
```

Note

In sample test, there are 4 paths with the sum of indices equal to 10:

H. New Year and Snowy Grid

time limit per test: 9 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Pay attention to the output section below, where you will see the information about flushing the output.

Bearland is a grid with h rows and w columns. Rows are numbered 1 through h from top to bottom. Columns are numbered 1 through w from left to right. Every cell is either allowed (denoted by '`.`' in the input) or permanently blocked (denoted by '`#`').

Bearland is a cold land, where heavy snow often makes travelling harder. Every day a few allowed cells are **temporarily** blocked by snow. Note, that this block works only on this particular day and next day any of these cells might be allowed again (unless there is another temporarily block).

It's possible to move directly between two cells only if they share a side and none of them is permanently or temporarily blocked.

Limak is a little polar bear who lives in Bearland. His house is at the top left cell, while his school is at the bottom right cell. Every day Limak should first go from his house to the school and then return back to his house. Since he gets bored easily, he doesn't want to **visit the same cell twice** on one day, except for the cell with his house, where he starts and ends. If Limak can reach a school and return home avoiding revisiting cells, he calls a day *interesting*.

There are q days you must process, one after another. For each of these days you should check if it's interesting and print "`YES`" or "`NO`" on a separate line. In order to be able to read the description of the next day you should print the answer for the previous one and flush the output.

It's guaranteed that a day with no cells temporarily blocked by snow would be interesting. It's also guaranteed that cells with Limak's house and school are never blocked (neither permanently or temporarily).

Input

The first line of the input contains three integers h, w and q ($2 \le h, w \le 1000$, $1 \le q \le 10\,000$) — the height and the width of the grid, and the number of days, respectively.

Next h lines describe which cells are allowed and which permanently blocked. The i-th line contains a string of length w, describing the i-th row. Every character is either '`.`' (denoting an allowed cell) or '`#`' (denoting a permanently blocked cell). It's guaranteed that a day with no cells temporarily blocked by snow would be interesting.

Then, the description of q days is given. The description of the i-th day starts with a line containing a single integer k_i ($1 \le k_i \le 10$) — the number of cells that are temporarily blocked by snow on that day. Each of next k_i lines contains two integers $r_{i,j}$ and $c_{i,j}$ ($1 \le r_{i,j} \le h$, $1 \le c_{i,j} \le w$), representing a cell at the intersection of the row $r_{i,j}$ and the column $c_{i,j}$. The given k_i cells are distinct and none of them is permanently blocked. Also, none of them contains Limak's house or school.

Output

For each of q days print "`YES`" if that day is interesting, and otherwise print "`NO`", both without the quotes. After printing an answer, you have to both print the end-of-line character and `flush` the output. Then you can proceed to the next day. You can get `Idleness Limit Exceeded` if you don't print anything or if you forget to flush the output.

To flush you can use (just after printing a `YES/NO` and end-of-line):

- `fflush(stdout)` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python;
- `flush(output)` in Pascal;
- See the documentation for other languages.

Examples

input

```
3 5 4
.....
.....
.#...
1
1 4
1
1 5
2
2 4
3 1
2
1 5
3 3
```

output

```
NO
YES
YES
NO
```

input

```
9 31 5
...............................
...............................
.###.###.#.###...###.###.#.###.
...#.#.#.#.#.......#.#.#.#...#.
.###.#.#.#.###...###.#.#.#...#.
.#...#.#.#.#.#...#...#.#.#...#.
.###.###.#.###...###.###.#...#.
...............................
...............................
5
6 5
2 11
1 14
8 15
2 14
5
2 14
1 14
8 16
6 5
2 11
3
2 2
1 4
8 30
10
3 1
3 11
5 16
7 21
4 16
3 5
7 31
3 9
7 25
3 27
10
3 1
3 9
7 25
3 27
7 21
4 17
3 5
7 31
4 16
3 11
```

output

```
NO
YES
YES
YES
NO
```

Note

In the first sample, there are 4 days. Drawings below show how Limak could go to school and return to his home in the second and the third day (on the left and on the right respectively). A permanently blocked cell is painted red, while cells temporarily blocked by snow are painted orange. Black and green arrows should Limak's way to the school and back to the house respectively.

For the second sample, below you can see how the grid looks like on each day, where '`#`' denotes a cell that is blocked, either temporarily or permanently.