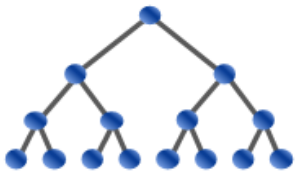


USACO 2017 JANUARY CONTEST, GOLD
PROBLEM 1. BALANCED PHOTO

Return to Problem List

Contest has ended.

Log in to allow submissions in analysis mode

English (en)

Farmer John is arranging his N cows in a line to take a photo ($1 \leq N \leq 100,000$). The height of the ith cow in sequence is h_i, and the heights of all cows are distinct.

As with all photographs of his cows, FJ wants this one to come out looking as nice as possible. He decides that cow i looks "unbalanced" if L_i and R_i differ by more than factor of 2, where L_i and R_i are the number of cows taller than i on her left and right, respectively. That is, i is unbalanced if the larger of L_i and R_i is strictly more than twice the smaller of these two numbers. FJ is hoping that not too many of his cows are unbalanced.

Please help FJ compute the total number of unbalanced cows.

INPUT FORMAT (file bphoto.in):

The first line of input contains N. The next N lines contain $h_1 \ldots h_N$, each a nonnegative integer at most 1,000,000,000.

OUTPUT FORMAT (file bphoto.out):

Please output a count of the number of cows that are unbalanced.

SAMPLE INPUT:

```
7
34
6
23
0
5
99
2
```

SAMPLE OUTPUT:

```
3
```

In this example, the cows of heights 34, 5, and 2 are unbalanced.

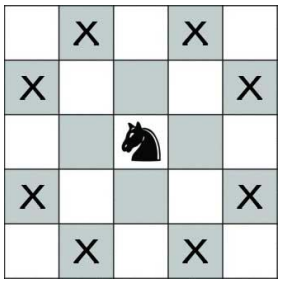
Problem credits: Brian Dean

Contest has ended. No further submissions allowed.

F: Knights

Magnus is the youngest chess grandmaster ever. He loves chess so much that he decided to decorate his home with chess pieces. To decorate his long corridor, he decided to use the knight pieces. His corridor is covered by beautiful square marble tiles of alternating colors, just like a chess board, with ***n*** rows and ***m*** columns. He will put images of knights on some (possibly none) of these tiles. Each tile will contain at most one knight.

The special thing about his arrangement is that there won't be any pair of knights can attack each other. Two knights can attack each other if they are placed in two opposite corner cells of a 2 by 3 rectangle. In this diagram, the knight can attack any of the Xs.



Given the dimension of the long corridor, your task is to calculate how many ways Magnus can arrange his knights. Two arrangements are considered different if there exists a tile which contains a knight in one arrangement but not in the other arrangement (in other words, rotations and reflections are considered different arrangements).

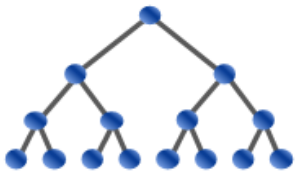
Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will consist of a single line with two integers ***n*** and ***m*** ($1 \le n \le 4$, $1 \le m \le 10^9$) representing the dimensions of the carpet. There will be a single space between ***n*** and ***m***.

Output

Output a single line with a single integer representing the number of possible arrangements, modulo (10^9+9). Output no spaces.

Sample Input	Sample Output
1 2	4
2 2	16
3 2	36



USACO 2017 DECEMBER CONTEST, PLATINUM
PROBLEM 2. PUSH A BOX

Return to Problem List

Contest has ended.

Log in to allow submissions in analysis mode

English (en)

Bessie and her friends have invented a new game. The game is named accurately, but not particularly creatively. They call it the "Push A Box Around The Barn To Get It In The Right Spot And Don't Move The Hay" game (if you think that's excessive, you should see some of the variable names the cows use when they write code...)

The barn can be modeled as an $N \times M$ rectangular grid. Some of the grid cells have hay in them. Bessie occupies one cell in this grid, and a large wooden box occupies another cell. Bessie and the box are not able to fit in the same cell at the same time, and neither can fit into a cell containing hay.

Bessie can move in the 4 orthogonal directions (north, east, south, west) as long as she does not walk into hay. If she attempts to walk to the space with the box, then the box will be pushed one space in that direction, as long as there is an empty cell on the other side. If there is no empty cell, then Bessie will not be able to make that move.

A certain grid cell is designated as the goal. Bessie's goal is to get the box into that location.

Given the layout of the barn, including the starting positions of the box and the cow, and the target position of the box, determine if it possible to win the game.

Note: This problem allows 512MB of memory usage, up from the default limit of 256MB.

INPUT FORMAT (file pushabox.in):

The first line has three numbers, N, M, and Q, where N is the number of rows in the grid and M is the number of columns.

- $1 \leq N, M \leq 1500$.
- $1 \leq Q \leq 50,000$.

On the next N lines is a representation of the grid, where characters represent empty cells (.), hay (#), Bessie's starting position (A), and the box's initial location (B).

This is followed by Q lines, each with a pair of integers (R, C). For each pair, you should determine if it is possible to get the box to that cell at row R, column C, starting from the initial state of the barn. The top row is row 1, and the left column is column 1.

OUTPUT FORMAT (file pushabox.out):

Q lines, each with either the string "YES" or "NO".

SAMPLE INPUT:

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

SAMPLE OUTPUT:

```
NO
YES
NO
NO
```

To push the box to the position (3, 5), the cow just needs to move 3 spaces to the right.

None of the other three positions are attainable.

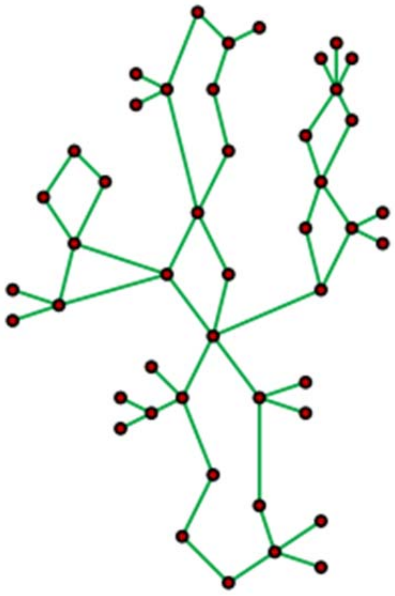
Problem credits: Nathan Pinsker

Contest has ended. No further submissions allowed.

UCF Local Contest — September 3, 2011

A Prickly Problem – Gold Edition

filename: `goldcactus`



A tree is a connected graph in which any two vertices have exactly one path between them.

A cactus (sometimes called a cactus tree) is a connected graph in which any two simple cycles have at most one vertex in common. Equivalently, every edge in such a graph belongs to at most one simple cycle. The graph pictured above is an example of a cactus graph.[1]

A spanning tree can be created from a connected graph by removing a set of edges such that if there are V vertices in the graph, then there are V – 1 edges remaining and every pair of vertices has exactly one path between them. Depending on which edges you choose to remove you will end up with different spanning trees. The cactus graph pictured above contains 36,864 spanning trees.

The Problem:

In this problem, your task is to count the number of spanning trees that a given cactus has. Since this result may be quite large, you should report your result modulo 1,007.

The Input:

Input will begin with an integer T denoting the number of test cases. Each test case will begin with two positive integers $V \leq 100$ and $E \leq (3*V)/2$ denoting the number of vertices and the number of edges, respectively. This will be followed by E lines, each containing an edge in the graph. Each edge is represented by its two vertices and each vertex listed will be between 1 and

[1] Picture of graph and definition of a cactus graph taken from www.wikipedia.org

V (assume that there is at most one edge in the input between any two vertices). It is guaranteed that the graph described in the input will be a cactus.

The Output:

For each test case, output a single line "Case #x: y" where x is the case number starting with 1 and y is the number of spanning trees modulo 1,007. Leave a blank line after the output for each test case. Follow the format illustrated in Sample Output.

Sample Input:

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

Sample Output:

```
Case #1: 3

Case #2: 9

Case #3: 1
```

UCF Local Contest — September 3, 2011

A Prickly Problem – Black Edition

filename: `blackcactus`

The Black Edition of Prickly Problem is identical to the Gold Edition except that it deals with a much larger input range (limit). More specifically, the new range for the number of vertices is:

$$V \leq 50{,}000$$

Please note that all the specs from the Gold Edition (except the number of vertices) should be followed for this Black Edition.

Illumination

Consider a square grid with lamps in fixed locations. Each lamp can either illuminate its row or its column, but not both. The illumination of each lamp extends over a limited distance.

Any square in the grid should only be illuminated by at most one lamp in its row and by at most one lamp in its column (one of each is acceptable, as is just the row, just the column, or neither). Determine if it is possible for all lamps to be lit while satisfying these constraints.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input contains three positive integers, ***n***, ***r*** and ***k*** (**1**≤***n***,***r***,***k***≤**1,000**, ***k***≤***n***x***n***), where ***n*** is the size of one side of the square grid, ***r*** is the maximum reach of a lamp, and ***k*** is the number of lamps. The next ***k*** lines will each contain two positive integers ***i*** and ***j*** (**1**≤***i***,***j***≤**n**), indicating that there is a lamp in the grid at row ***i***, column ***j***.

Each lamp can illuminate squares at most ***r*** units away, and can also illuminate its own square, so the maximum number of squares it can illuminate is **2*r*+1**. All lamps will be in distinct locations.

Output

Output a single integer, **1** if it is possible to light all of the lamps and **0** if it is not possible.

Sample Input	Sample Output
3 2 5 1 1 1 3 3 1 3 3 2 2	1
3 2 6 1 1 1 2 1 3 3 1 3 2 3 3	0

Problem K
Kindergarten
Problem ID: kindergarten

Photo by Lars Plougmann

Every year the three friendly teachers at the kindergarten let their classes' kids change classes to form three new ones. Some kids, of course, are old enough to leave, but those who stay for another year are rearranged among the three teachers.

The teachers even let the kids have their say in the process. As friendship both comes and goes hastily in young years, each kid X ranks every other kid Y according to how glad X would be to have Y in her new class. In fact, X produces a preference list giving a total order of the other kids, i.e. there are no such things as ties – kids she would be equally glad to have as class mates.

The three teachers do not mind if the new classes formed are unbalanced in size, since they fill up their classes with new kids about to start their first year at the kindergarten. They do, however, want all the kids in their own new class to be different from last year, since even a teacher needs a break after a whole year with the same kids. They decide that a best partition into three classes under these premises is one where no kid is in the same class as a kid not listed among the top T entries on their preference list, for T as small as possible. Note that the kids in a new class may very well be the same as in an old one, but then with a new teacher!

Input

The first line of input contains a positive integer $n \leq 200$ giving the number of kids to be rearranged at the kindergarden. The kids are numbered 1 through n.
Then follow n lines describing the kids. The i-th row first contains the identifier of their current class' teacher (an integer 0, 1, or 2), and next the $n-1$ integers $\{1, 2, 3, \ldots, i-1, i+1, \ldots, n\}$ in some order, describing the class mate preference list of the i-th kid, in descending order.

Output

The smallest non-negative integer T, such that there is a partitioning of the kids in three new classes such that

- No kid has the same teacher as in their current class, and
- all kids' class mates are among the top T places of their preference lists, respectively.

Sample Input 1

```
6
0 2 3 4 5 6
0 1 3 4 5 6
1 6 5 4 2 1
2 6 5 3 2 1
1 1 2 3 4 6
2 1 2 3 4 5
```

Sample Output 1

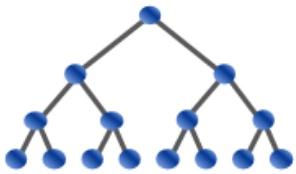
```
4
```

Sample Input 2

```
3
0 2 3
1 1 3
2 1 2
```

Sample Output 2

```
0
```



USACO 2015 JANUARY CONTEST, GOLD
PROBLEM 3. GRASS COWNOISSEUR

Return to Problem List

Contest has ended.

Log in to allow submissions in analysis mode

English (en)

Problem 3: Grass Cownoisseur [Mark Gordon, 2015]

In an effort to better manage the grazing patterns of his cows, Farmer John has installed one-way cow paths all over his farm. The farm consists of N fields, conveniently numbered 1..N, with each one-way cow path connecting a pair of fields. For example, if a path connects from field X to field Y, then cows are allowed to travel from X to Y but not from Y to X.

Bessie the cow, as we all know, enjoys eating grass from as many fields as possible. She always starts in field 1 at the beginning of the day and visits a sequence of fields, returning to field 1 at the end of the day. She tries to maximize the number of distinct fields along her route, since she gets to eat the grass in each one (if she visits a field multiple times, she only eats the grass there once).

As one might imagine, Bessie is not particularly happy about the one-way restriction on FJ's paths, since this will likely reduce the number of distinct fields she can possibly visit along her daily route. She wonders how much grass she will be able to eat if she breaks the rules and follows up to one path in the wrong direction. Please compute the maximum number of distinct fields she can visit along a route starting and ending at field 1, where she can follow up to one path along the route in the wrong direction. Bessie can only travel backwards at most once in her journey. In particular, she cannot even take the same path backwards twice.

INPUT: (file grass.in)

The first line of input contains N and M, giving the number of fields and the number of one-way paths (1 <= N, M <= 100,000).

The following M lines each describe a one-way cow path. Each line contains two distinct field numbers X and Y, corresponding to a cow path from X to Y. The same cow path will never appear more than once.

SAMPLE INPUT:

```
7 10
1 2
3 1
2 5
2 4
3 7
3 5
3 6
6 5

7 2
4 7
```

OUTPUT: (file grass.out)

A single line indicating the maximum number of distinct fields Bessie

```
can visit along a route starting and ending at field 1, given that she can
follow at most one path along this route in the wrong direction.

SAMPLE OUTPUT: (file grass.out)

6

SOLUTION NOTES:

Here is an ASCII drawing of the sample input:

v---3-->6
7   |\  |
^\  v \ |
| \ 1  \|
|  \|   v
|   v   5
4<--2---^

Bessie can visit pastures 1, 2, 4, 7, 2, 5, 3, 1 by traveling
backwards on the path between 5 and 3.  When she arrives at 3 she
cannot reach 6 without following another backwards path.
```

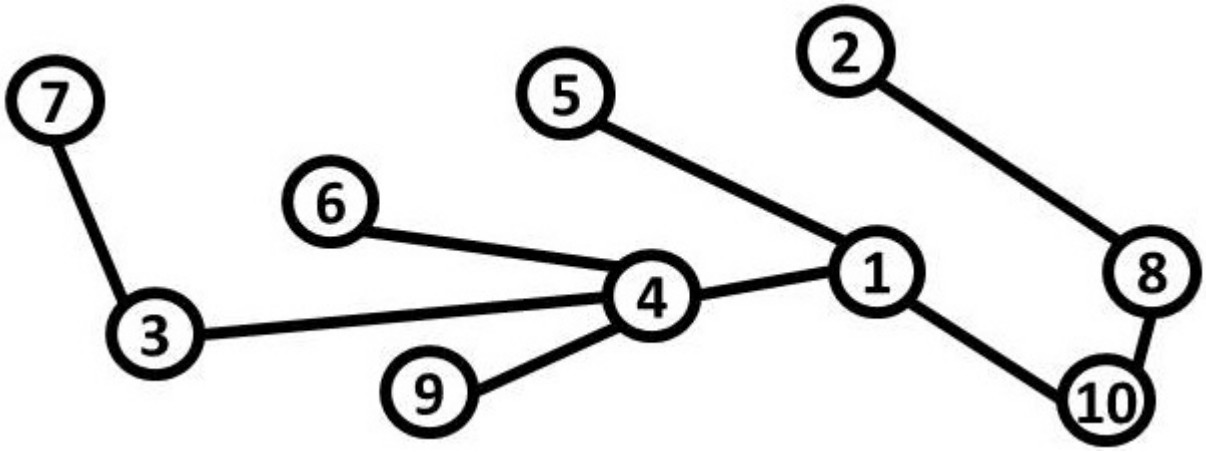
Contest has ended. No further submissions allowed.

Problem I
Tourists

In Tree City, there are n tourist attractions uniquely labeled 1 to n. The attractions are connected by a set of $n-1$ bidirectional roads in such a way that a tourist can get from any attraction to any other using some path of roads.

You are a member of the Tree City planning committee. After much research into tourism, your committee has discovered a very interesting fact about tourists: they LOVE number theory! A tourist who visits an attraction with label x will then visit another attraction with label y if $y > x$ and y is a multiple of x. Moreover, if the two attractions are not directly connected by a road the tourist will necessarily visit all of the attractions on the path connecting x and y, even if they aren't multiples of x. The number of attractions visited includes x and y themselves. Call this the *length* of a path.

Consider this city map:



Here are all the paths that tourists might take, with the lengths for each:
$1 \to 2 = 4$, $1 \to 3 = 3$, $1 \to 4 = 2$, $1 \to 5 = 2$, $1 \to 6 = 3$, $1 \to 7 = 4$,
$1 \to 8 = 3$, $1 \to 9 = 3$, $1 \to 10 = 2$, $2 \to 4 = 5$, $2 \to 6 = 6$, $2 \to 8 = 2$,
$2 \to 10 = 3$, $3 \to 6 = 3$, $3 \to 9 = 3$, $4 \to 8 = 4$, $5 \to 10 = 3$

To take advantage of this phenomenon of tourist behavior, the committee would like to determine the number of attractions on paths from an attraction x to an attraction y such that $y > x$ and y is a multiple of x. You are to compute the **sum** of the lengths of all such paths. For the example above, this is: $4+3+2+2+3+4+3+3+2+5+6+2+3+3+3+4+3 = 55$.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input will consist of an integer n ($2 \leq n \leq 200{,}000$) indicating the number of attractions. Each of the following $n-1$ lines will consist of a pair of space-separated integers i and j ($1 \leq i < j \leq n$), denoting that attraction i and attraction j are directly connected by a road. It is guaranteed that the set of attractions is connected.

Output

Output a single integer, which is the sum of the lengths of all paths between two attractions x and y such that $y > x$ and y is a multiple of x.

Sample Input 1	Sample Output 1
10 3 4 3 7 1 4 4 6 1 10 8 10 2 8 1 5 4 9	55

USA Computing Olympiad

USACO 2018 FEBRUARY CONTEST, PLATINUM
PROBLEM 2. NEW BARNS

Return to Problem List

Contest has ended.

Log in to allow submissions in analysis mode

English (en)

Farmer John notices that his cows tend to get into arguments if they are packed too closely together, so he wants to open a series of new barns to help spread them out.

Whenever FJ constructs a new barn, he connects it with at most one bidirectional pathway to an existing barn. In order to make sure his cows are spread sufficiently far apart, he sometimes wants to determine the distance from a certain barn to the farthest possible barn reachable from it (the distance between two barns is the number of paths one must traverse to go from one barn to the other).

FJ will give a total of Q ($1 \leq Q \leq 10^5$) queries, each either of the form "build" or "distance". For a build query, FJ builds a barn and links it with at most one previously built barn. For a distance query, FJ asks you the distance from a certain barn to the farthest barn reachable from it via a series of pathways. It is guaranteed that the queried barn has already been built. Please help FJ answer all of these queries.

INPUT FORMAT (file newbarn.in):

The first line contains the integer Q. Each of the next Q lines contains a query. Each query is of the form "B p" or "Q k", respectively telling you to build a barn and connect it with barn p, or give the farthest distance, as defined, from barn k. If $p = -1$, then the new barn will be connected to no other barn. Otherwise, p is the index of a barn that has already been built. The barn indices start from 1, so the first barn built is barn 1, the second is barn 2, and so on.

OUTPUT FORMAT (file newbarn.out):

Please write one line of output for each distance query. Note that a barn which is connected to no other barns has farthest distance 0.

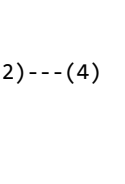
SAMPLE INPUT:

```
7
B -1
Q 1
B 1
B 2
Q 3
B 2
Q 2
```

SAMPLE OUTPUT:

```
0
2
1
```

The example input corresponds to this network of barns:

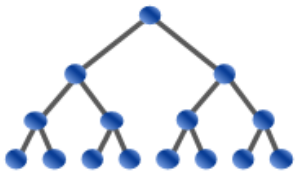


In query 1, we build barn number 1. In query 2, we ask for the distance of 1 to the farthest connected barn. Since barn 1 is connected to no other barns, the answer is 0. In query 3, we build barn number 2 and connect it to barn 1. In query 4, we build barn number 3 and connect it to barn 2. In query 5, we ask for the distance of 3 to the farthest connected barn. In this case, the

farthest is barn 1, which is 2 units away. In query 6, we build barn number 4 and connect it to barn 2. In query 7, we ask for the distance of 2 to the farthest connected barn. All three barns 1, 3, 4 are the same distance away, which is 1, so this is our answer.

Problem credits: Anson Hu

Contest has ended. No further submissions allowed.



USACO 2018 US OPEN CONTEST, PLATINUM
PROBLEM 3. DISRUPTION

Return to Problem List

Contest has ended.

Log in to allow submissions in analysis mode

English (en)

Farmer John prides himself on running a well-connected farm. The farm is a collection of N pastures ($2 \leq N \leq 50,000$), pairs of which are connected with $N-1$ bi-directional pathways, each having unit length. Farmer John notices that using an appropriate series of these pathways, it is possible to travel from any pasture to any other pasture.

Although FJ's farm is connected, he worries what might happen if one of the pathways gets blocked, as this would effectively partition his farm into two disjoint sets of pastures, where the cows could travel within each set but not between the sets. FJ therefore builds a set of M additional bi-directional pathways ($1 \leq M \leq 50,000$), each with a positive integer length at most 10^9. The cows still only use the original pathways for transit, unless one of these becomes blocked.

If one of the original pathways becomes blocked, the farm becomes partitioned into two disjoint pieces, and FJ will select from among his extra pathways a single replacement pathway that re-establishes connectivity between these two pieces, so the cows can once again travel from any pasture to any other pasture.

For each of the original pathways on the farm, help FJ select the shortest suitable replacement pathway.

INPUT FORMAT (file disrupt.in):

The first line of input contains N and M. Each of the next $N-1$ lines describes an original pathway using integers p, q, where $p \neq q$ are the pastures connected by the pathway (in the range $1 \ldots N$). The remaining M lines each describe an extra pathway in terms of three integers: p, q, and r, where r is the length of the pathway. At most one pathway runs between any pair of pastures.

OUTPUT FORMAT (file disrupt.out):

For each of the $N-1$ original pathways in the order they appear in the input, output the length of the shortest suitable replacement pathway which would re-connect the farm if that original pathway were to be blocked. If no suitable replacement exists, output -1.

SAMPLE INPUT:

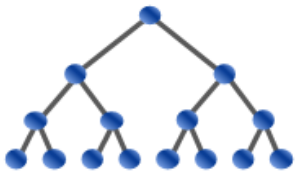
```
6 3
1 2
1 3
4 1
4 5
6 5
2 3 7
3 6 8
6 4 5
```

SAMPLE OUTPUT:

```
7
7
8
5
5
```

Problem credits: Brian Dean

Contest has ended. No further submissions allowed.



USACO 2017 US OPEN CONTEST, PLATINUM
PROBLEM 3. COWBASIC

Return to Problem List

Contest has ended.

Log in to allow submissions in analysis mode

English (en)

Bessie has invented a new programming language, but since there is no compiler yet, she needs your help to actually run her programs.

COWBASIC is a simple, elegant language. It has two key features: addition and MOO loops. Bessie has devised a clever solution to overflow: all addition is done modulo $10^9 + 7$. But Bessie's real achievement is the MOO loop, which runs a block of code a fixed number of times. MOO loops and addition can, of course, be nested.

Given a COWBASIC program, please help Bessie determine what number it returns.

INPUT FORMAT (file cowbasic.in):

You are given a COWBASIC program at most 100 lines long, with each line being at most 350 characters long. A COWBASIC program is a list of statements.

There are three types of statements:

```
<variable> = <expression>

<literal> MOO {
  <list of statements>
}

RETURN <variable>
```

There are three types of expressions:

```
<literal>

<variable>

( <expression> ) + ( <expression> )
```

A literal is a positive integer at most 100,000.

A variable is a string of at most 10 lowercase English letters.

It is guaranteed that no variable will be used or RETURNed before it is defined. It is guaranteed that RETURN will happen exactly once, on the last line of the program.

OUTPUT FORMAT (file cowbasic.out):

Output a single positive integer, giving the value of the RETURNed variable.

Scoring

- In 20 percent of all test cases - MOO loops are not nested.
- In another 20 percent of all test cases - The program only has 1 variable. MOO loops can be nested.
- In the remaining test cases, there are no further restrictions.

SAMPLE INPUT:

```
x = 1
10 MOO {
  x = ( x ) + ( x )
}
```

```
RETURN x
```

SAMPLE OUTPUT:

```
1024
```

This COWBASIC program computes 2^{10}.

SAMPLE INPUT:

```
n = 1
nsq = 1
100000 MOO {
  100000 MOO {
    nsq = ( nsq ) + ( ( n ) + ( ( n ) + ( 1 ) ) )
    n = ( n ) + ( 1 )
  }
}
RETURN nsq
```

SAMPLE OUTPUT:

```
4761
```

This COWBASIC program computes $(10^5 * 10^5 + 1)^2$ (modulo $10^9 + 7$).

Problem credits: Jonathan Paulson

Contest has ended. No further submissions allowed.

Problem I: Zigzag Subsequence

Filename: `zigzag2`

Timelimit: `20 seconds`

Zigzag sequences are a curious kind of integer sequence. They are sequences of ***at least length 3*** that have the following form every three consecutive integers a_i, a_{i+1}, a_{i+2}:

$a_i < a_{i+1} > a_{i+2}$ or $a_i > a_{i+1} < a_{i+2}$

Examples of zigzag sequences are 14, 17, 3, 19 and 1, 7, 1, 97, 2. Examples of sequences that are not zigzag sequences are 1, 17, 29, 17, 1 and 1, 9, 9.

Given a sequence, you are to find how many subsequences form a zigzag sequence. A subsequence is formed by removing some number, possibly zero, of integers from the sequence. A subsequence is considered different if the i^{th} location is removed in one sequence but not the other.

Input

The first line contains a positive integer ***s*** (***s*** ≤ 40), representing the number of sequences to consider.

The next r lines contains a positive integer ***n***, ($3 \le n \le 10^5$) representing the length of the original sequence.

The following line contains ***n*** integers. Each integer a_i ($-10^9 \le a_i \le 10^9$), represents the i^{th} integer in the sequence.

Output

For each sequence, output a single integer on a line by itself representing the number of subsequences that are also zigzag sequences. Since this number can be quite large, output this value modulo 10^9+7.

Samples

Input	Output
3 3 -7 8 -8 4 1 4 1 4 5 1 4 1 4 1	1 3 8

UCF Local Contest — September 2, 2017

Rotating Cards

filename: `cards`
(*Difficulty Level:* `Medium/Hard`)

A magician has a stack of *n* cards labeled 1 through *n*, in random order. Her trick involves discarding all of the cards in numerical order (first the card labeled 1, then the card labeled 2, etc.). Unfortunately, she can only discard the card on the top of her stack and the only way she can change the card on the top of her stack is by moving the bottom card on the stack to the top, or moving the top card on the stack to the bottom. The cost of moving any card from the top to the bottom or vice versa is simply the value of the label on the card. There is no cost to discard the top card of the stack. Help the magician calculate the minimum cost for completing her trick.

The Problem:

Given the number of cards in the magician's stack and the order of those cards in the stack, determine the minimum cost for her to discard all of the cards.

The Input:

The first input line contains a positive integer, *t*, indicating the number of test cases to process. Each test case is on a separate input line by itself and starts with an integer, *c* ($1 \leq c \leq 10^5$), indicating the number of cards in the stack, followed by *c* labels for the cards in the stack (starting from the top going to the bottom). Each of these labels will be in between 1 and *c*, inclusive, and each label will be unique.

The Output:

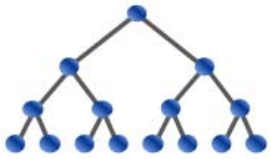
For each test case, output a single integer on a line by itself indicating the minimum cost for the magician to complete her magic trick.

Sample Input:

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

Sample Output:

```
15
0
```



USACO 2015 DECEMBER CONTEST, PLATINUM
PROBLEM 3. COUNTING HAYBALES

Return to Problem List

Contest has ended.

Log in to allow submissions in analysis mode

English (en)

Farmer John is trying to hire contractors to help rearrange his farm, but so far all of them have quit when they saw the complicated sequence of instructions FJ wanted them to follow. Left to complete the project by himself, he realizes that indeed, he has made the project perhaps more complicated than necessary. Please help him follow his instructions to complete the farm upgrade.

FJ's farm consists of N fields in a row, conveniently numbered $1 \ldots N$. In each field there can be any number of haybales. Farmer John's instructions contain three types of entries:

1) Given a contiguous interval of fields, add a new haybale to each field.

2) Given a contiguous interval of fields, determine the minimum number of haybales in a field within that interval.

3) Given a contiguous interval of fields, count the total number of haybales inside that interval.

INPUT FORMAT (file haybales.in):

The first line contains two positive integers, N ($1 \leq N \leq 200,000$) and Q ($1 \leq Q \leq 100,000$).

The next line contains N nonnegative integers, each at most 100,000, indicating how many haybales are initially in each field.

Each of the next Q lines contains a single uppercase letter, either M, P or S, followed by either two positive integers A and B ($1 \leq A \leq B \leq N$), or three positive integers A, B, and C ($1 \leq A \leq B \leq N$; $1 \leq C \leq 100,000$). There will be three positive integers if and only if the uppercase letter is P.

If the letter is M, print the minimum number of haybales in the interval of fields from $A \ldots B$.

If the letter is P, put C new haybales in each field in the interval of fields from $A \ldots B$.

If the letter is S, print the total number of haybales found within interval of fields from $A \ldots B$.

OUTPUT FORMAT (file haybales.out):

A line in the output should appear in response to every 'M' or 'S' entry in FJ's instructions.

SAMPLE INPUT:

```
4 5
3 1 2 4
M 3 4
S 1 3
P 2 3 1
M 3 4
S 1 3
```

SAMPLE OUTPUT:

```
2
6
3
8
```

Problem credits: Nick Wu

Problem H: Star-Belly Sneetches

Filename: `sneetches`
Timelimit: `30 seconds`

Now, the Star-Belly Sneetches had bellies with stars.
The Plain-Belly Sneetches had none upon thars.
Those stars weren't so big. They were really so small.
You might think such a thing wouldn't matter at all.

"My friends", he announced in a voice clear and clean,
"My name is Sylvester McMonkey McBean.
And I've heard of Your troubles. I've heard you're unhappy.
But I can fix that, I'm the Fix-It-Up Chappie.

Then, quickly, Sylvester McMonkey McBean
Put together a very peculiar machine.
And he said, "You want stars like a Star-Belly Sneetch?
My friends, you can have them for three dollars each!"

Then, of course, those with stars got all frightfully mad.
To be wearing a star was frightfully bad.
Then, of course, old Sylvester McMonkey McBean
invited THEM into his Star-Off Machine.

Then, of course from THEN on, as you probably guess,
Things really got into a horrible mess.

Now Sylvester McMonkey McBean is using his Star-Swapping machine to add and remove stars from Sneetches that want them or don't want them. The Sneetches keep wanting to either add a star to their belly or remove the star from their belly. Sylvester now wants to keep track of which sneetches on the beaches are most likely to want their star swapped next.

He does this by observing that the sneetches are lined up on the beaches in order from left to right. He can label the sneetches using the numbers 1 to ***n***, where ***n*** is the number of sneetches. Sylvester knows that each Sneetch is more likely to swap stars if those around him are the same type of sneetch. The Sneetches want to be different! That is why he wants to know the longest contiguous block of Star-Bellied Sneetches and the longest contiguous block of Plain-Bellied Sneetches. As the swapping happens things may change. That is why he needs to know these values when some range of consecutive sneetches choose to swap their stars.

Input

The first line contains a positive integer ***t*** (***t*** ≤ 50), representing the number of beaches with Sneetches to consider.

For each town, the first line contains two positive integers ***n*** and ***s***, (1 ≤ ***n***, ***s*** ≤ 10^5) representing the number of Sneetches on the beaches and the number of star swapping operations to perform.

The following line is a string of ***n*** characters. Each character is either 'S' or 'P', representing if the i^{th} sneetch on the beach is a Star-Bellied Sneetch or a Plain-Bellied Sneetch.

The next line ***s*** lines contain two integers each ***a*** and ***b***, (1 ≤ ***a*** ≤ ***b*** ≤ ***n***) representing the interval of sneetches on which to perform the star swapping. The range [***a***, ***b***] is inclusive.

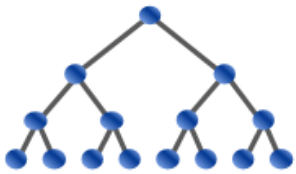
Output

For each star swapping, output two integers on a line by themselves. The first represents the largest consecutive group of star-bellied sneetches and the second represents the largest consecutive group of plain-bellied sneetches.

Samples

Input	Output
3 4 2 SSSS 2 3 3 4 4 1 SPPP 2 4 6 5 PPPPPP 1 3 2 4 3 5 4 4 6 6	1 2 1 1 4 0 3 3 1 2 1 1 3 1 4 1

The annotations below show how the sneetches change after each swap operation.
Case 1: SSSS -> SPPS -> SPSP
Case 2: SPPP -> SSSS
Case 3: PPPPPP -> SSSPPP -> SPPSPP -> SPSPSP -> SPSSSP -> SPSSSS



USACO 2015 FEBRUARY CONTEST, SILVER
PROBLEM 1. CENSORING (SILVER)

Return to Problem List

Contest has ended.

Log in to allow submissions in analysis mode

English (en)

Farmer John has purchased a subscription to Good Hooveskeeping magazine for his cows, so they have plenty of material to read while waiting around in the barn during milking sessions. Unfortunately, the latest issue contains a rather inappropriate article on how to cook the perfect steak, which FJ would rather his cows not see (clearly, the magazine is in need of better editorial oversight).

FJ has taken all of the text from the magazine to create the string S of length at most 10^6 characters. From this, he would like to remove occurrences of a substring T to censor the inappropriate content. To do this, Farmer John finds the _first_ occurrence of T in S and deletes it. He then repeats the process again, deleting the first occurrence of T again, continuing until there are no more occurrences of T in S. Note that the deletion of one occurrence might create a new occurrence of T that didn't exist before.

Please help FJ determine the final contents of S after censoring is complete.

INPUT FORMAT: (file censor.in)

The first line will contain S. The second line will contain T. The length of T will be at most that of S, and all characters of S and T will be lower-case alphabet characters (in the range a..z).

OUTPUT FORMAT: (file censor.out)

The string S after all deletions are complete. It is guaranteed that S will not become empty during the deletion process.

SAMPLE INPUT:

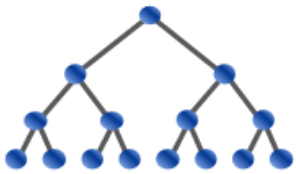
```
whatthemomooofun
moo
```

SAMPLE OUTPUT:

```
whatthefun
```

[Problem credits: Mark Gordon, 2015]

Contest has ended. No further submissions allowed.



USACO 2015 FEBRUARY CONTEST, GOLD
PROBLEM 2. CENSORING (GOLD)

Return to Problem List

Contest has ended.

Log in to allow submissions in analysis mode

English (en)

Farmer John has purchased a subscription to Good Hooveskeeping magazine for his cows, so they have plenty of material to read while waiting around in the barn during milking sessions. Unfortunately, the latest issue contains a rather inappropriate article on how to cook the perfect steak, which FJ would rather his cows not see (clearly, the magazine is in need of better editorial oversight).

FJ has taken all of the text from the magazine to create the string S of length at most 10^5 characters. He has a list of censored words t_1 ... t_N that he wishes to delete from S. To do so Farmer John finds the earliest occurrence of a censored word in S (having the earliest start index) and removes that instance of the word from S. He then repeats the process again, deleting the earliest occurrence of a censored word from S, repeating until there are no more occurrences of censored words in S. Note that the deletion of one censored word might create a new occurrence of a censored word that didn't exist before.

Farmer John notes that the censored words have the property that no censored word appears as a substring of another censored word. In particular this means the censored word with earliest index in S is uniquely defined.

Please help FJ determine the final contents of S after censoring is complete.

INPUT FORMAT: (file censor.in)

The first line will contain S. The second line will contain N, the number of censored words. The next N lines contain the strings t_1 ... t_N. Each string will contain lower-case alphabet characters (in the range a..z), and the combined lengths of all these strings will be at most 10^5.

OUTPUT FORMAT: (file censor.out)

The string S after all deletions are complete. It is guaranteed that S will not become empty during the deletion process.

SAMPLE INPUT:

```
begintheescapexecutionatthebreakofdawn
2
escape
execution
```

SAMPLE OUTPUT:

```
beginthatthebreakofdawn
```

[Problem credits: Mark Gordon, 2015]

Contest has ended. No further submissions allowed.

The Big Painting

Samuel W. E. R. Craft is an artist with a growing reputation. Unfortunately, the paintings he sells do not provide him enough money for his daily expenses plus the new supplies he needs. He had a brilliant idea yesterday when he ran out of blank canvas: "Why don't I create a gigantic new painting, made of all the unsellable paintings I have, stitched together?". After a full day of work, his masterpiece was complete.

That's when he received an unexpected phone call: a client saw a photograph of one of his paintings and is willing to buy it now! He had forgotten to tell the art gallery to remove his old works from the catalog! He would usually welcome a call like this, but how is he going to find his old work in the huge figure in front of him?

Galerie de Vues de la Rome Moderne, Panini (1759). What S.W.E.R.C. had in mind when he tried to merge his paintings.

Task

Given a black-and-white representation of his original painting and a black-and-white representation of his masterpiece, can you help S.W.E.R.C. identify in how many locations his painting might be?

Input

The first line consists of 4 space-separated integers: h_p w_p h_m w_m, the height and width of the painting he needs to find, and the height and width of his masterpiece, respectively.

The next h_p lines have w_p lower-case characters representing his painting. After that, the next h_m lines have w_m lower-case characters representing his masterpiece. Each character will be either 'x' or 'o'.

Constraints

$1 \leq h_p, w_p \leq 2\,000$
$1 \leq h_m, w_m \leq 2\,000$
$h_p \leq h_m$
$w_p \leq w_m$

Output

A single integer representing the number of possible locations where his painting might be.

Sample Input

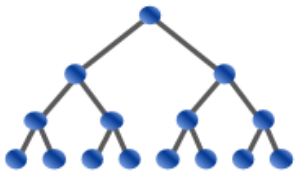
```
4 4 10 10
oxxo
xoox
xoox
oxxo
xxxxxxoxxo
oxxoooxoox
xooxxxxoox
xooxxxoxxo
oxxoxxxxxx
ooooxxxxxx
xxxoxxoxxo
oooxooxoox
oooxooxoox
xxxoxxoxxo
```

Sample Output

```
4
```

Sample Output Explanation

The painting could be in four locations as shown in the following picture. Two of the locations overlap.



USACO 2017 DECEMBER CONTEST, PLATINUM
PROBLEM 1. STANDING OUT FROM THE HERD

Return to Problem List

Contest has ended.

Log in to allow submissions in analysis mode

English (en)

Just like humans, cows often appreciate feeling they are unique in some way. Since Farmer John's cows all come from the same breed and look quite similar, they want to measure uniqueness in their names.

Each cow's name has some number of substrings. For example, "amy" has substrings {a, m, y, am, my, amy}, and "tommy" would have the following substrings: {t, o, m, y, to, om, mm, my, tom, omm, mmy, tomm, ommy, tommy}.

A cow name has a "uniqueness factor" which is the number of substrings of that name not shared with any other cow. For example, If amy was in a herd by herself, her uniqueness factor would be 6. If tommy was in a herd by himself, his uniqueness factor would be 14. If they were in a herd together, however, amy's uniqueness factor would be 3 and tommy's would be 11.

Given a herd of cows, please determine each cow's uniqueness factor.

INPUT FORMAT (file standingout.in):

The first line of input will contain N ($1 \leq N \leq 10^5$). The following N lines will each contain the name of a cow in the herd. Each name will contain only lowercase characters a-z. The total length of all names will not exceed 10^5.

OUTPUT FORMAT (file standingout.out):

Output N numbers, one per line, describing the uniqueness factor of each cow.

SAMPLE INPUT:

```
3
amy
tommy
bessie
```

SAMPLE OUTPUT:

```
3
11
19
```

Problem credits: Matt Fontaine

Contest has ended. No further submissions allowed.

At Least Average

Filename: *average*
Time Limit: *8 seconds*

Blake plays a video game that has *n* levels. On each level, she can earn in between 0 and 10 points, inclusive. The number of points she gets on a level must be an integer. She has set up a goal for herself to equal or beat a particular average. In order to make her performance seem impressive, she's decided that she wants to count the number of intervals (sets of consecutive levels) where she's equaled or beaten her target average.

We define an interval [i, j], with $1 \leq i \leq j \leq n$, to be each level starting at level i and ending at level j, including level j. For example, if Blake played 5 levels with her scores being 5, 1, 2, 4 and 6, and Blake's target average was 3, then here are the following intervals where she equaled or beat her target average:

[1, 1] with average 5
[1, 2] with average 3
[1, 4] with average 3
[1, 5] with average 3.6
[2, 5] with average 3.25
[3, 4] with average 3
[3, 5] with average 4
[4, 4] with average 4
[4, 5] with average 5
[5, 5] with average 6

Blake can make the impressive statement that on 10 intervals her average score was 3 or more.

The Problem

Given the number of levels Blake has played in her video game, her scores on each level, and the average value she'd like to equal or beat, determine the number of intervals that she equaled or beat the given average.

The Input

The first line of input will consist of a single positive integer, *v* ($v \leq 10$), representing the number of input cases to process. Input for each case follows, one case taking two lines. The first line of input for each case contains two positive integers, *n* ($n \leq 100000$), and *a* ($a \leq 10$), separated by spaces, representing the number of levels of the video game and the average Blake wants to obtain, respectively. The following line contains *n* space separated integers, each in between 0 and 10 inclusive, representing Blake's score on each of the *n* levels, in order.

The Output

For each input case, output a single integer on a line by itself representing the number of intervals where Blake obtained her desired average or higher.

Sample Input

```
2
5 3
5 1 2 4 6
9 6
10 1 10 1 10 1 10 1 10
```

Sample Output

```
10
15
```

Problem B
Bread Sorting
Problem ID: bread

Photo by Jim Champion

Michael works in a bakery. At the end of his shift, his boss wants the breads sorted in a certain order. She can't seem to decide on which order, though – every day there seems to be a new one – to Michael's despair. Michael has worked there for a while now and has learned a neat trick with his wooden bakery paddle. He can take three breads next to each other on his paddle and throw them up in the air such that when they land, the right-most has moved to the left-most position, and the two other breads have moved one place to the right. In other words, he can rotate to the right a subsequence of breads of length three.

Before the end of the shift, his coworkers place the breads in a long line. Michael would like to sort the line of breads using his paddle trick. He can take any three consecutive breads along the line on his paddle, rotate them, and then put them back. Sometimes though, it does not matter how many times he uses his paddle – the line of bread just doesn't seem to be possible to sort the way the boss wants...

Input

The first line of input contains a positive integer n, $3 \leq n \leq 100\,000$. Then two lines follow: On the first line, a permutation of the integers 1 through n describing the order in which the breads are lined up. On the second line, a permutation of the integers 1 through n describing how Michael's boss wants the breads sorted.

Output

Output "`Possible`" if Michael can sort the breads with his paddle in the order prescribed by his boss, otherwise "`Impossible`".

Sample Input 1	Sample Output 1
4 1 3 4 2 4 3 2 1	Possible

Sample Input 2	Sample Output 2
6 1 2 3 4 5 6 6 5 4 3 2 1	Impossible

Problem B: Patience

Patience or solitaire is the name for a group of single player card games. One new such game, so new it has no name, is played with cards sporting random integers as values. The game starts by shuffling all cards and distributing them in **N** sequences, not necessarily of equal length.

During each turn, the player can remove the first card in any sequence and place it at the end of the *solution sequence*. The card that was second in the selected sequence now becomes the first and the turn ends. Of course, once the card is in the *solution sequence* it cannot be removed, replaced or altered in any way. So don't even try.

The game ends when all cards are in the *solution sequence*. The object of the game is to construct the best possible *solution sequence*. One sequence is better than the other if, for the first cards they differ; let's call them X and Y; the value on the card X is smaller than the value on the card Y.

Write a program that finds the best possible *solution sequence*.

Input

The first line contains one integer **N** ($1 \leq \mathbf{N} \leq 1000$), number of starting sequences. The next **N** lines contain the description of input sequences. Each line starts with an integer **L** ($1 \leq \mathbf{L} \leq 1000$), the length of the sequence. It is followed by **L** integers $\mathbf{v_i}$ ($1 \leq \mathbf{v_i} \leq 10^8$).

Output

Output one line containing $\sum$ **L** numbers, the best possible *solution sequence* obtainable.

Samples

Input	Output
3 1 2 1 100 1 1	1 2 100
2 5 10 20 30 40 50 2 28 27	10 20 28 27 30 40 50
2 3 5 1 2 3 5 1 1	5 1 1 5 1 2