1. Let’s Sort this Out

PROBLEM

The classic sorting problem! Given a list of triplet values (a boolean, a string, and an integer), sort and print out the list. The sort order is as follows:

1. BOOLEAN: FALSE -> TRUE
2. STRING: LOWEST ALPHA -> HIGHEST ALPHA
3. INTEGER: LOWEST -> HIGHEST

INPUT

The first line contains **N**, the number of triplets.
The next **N** lines contains a triplet.

CONSTRAINTS

N < 1000.

OUTPUT

Output the list, but sorted.

SAMPLE TEST CASE 1:
INPUT:

```
8
TRUE bb 2
FALSE aa 6
TRUE ab 9
FALSE ba 3
FALSE ab 3
FALSE aa 4
TRUE aa 5
FALSE bb 8
```

OUTPUT:

```
FALSE aa 4
FALSE aa 6
FALSE ab 3
FALSE ba 3
FALSE bb 8
TRUE aa 5
TRUE ab 9
TRUE bb 2
```

2. Bad Addition

PROBLEM

Your job is to add up a bunch of numbers! However, each number is in a specified base. Each number will be written as “[base]b[number]”. However, some of the numbers will be messed up! They may have the wrong format, not be a real number for that base, or something else entierly. The highest valid base will be 62.

All you have to do is output the sum of the valid numbers, in base 10.

INPUT

The first line contains **N**, the number of integers.
The next **N** lines contain the numbers to add.

CONSTRAINTS

N < 1000

OUTPUT

The sum of the valid numbers, in base 10.

SAMPLE TEST CASE 1:

INPUT:

```
13
2b1001
2b1421
hello how are you doing?
5512
6672b
3b210
8b
6bdonut
5
5b 104
64b102
62bT
10b40
```

OUTPUT:

```
106
```

3. What’d I Miss?

PROBLEM

A friend tried to generate you a list of integers from 1 to **N**, but used Python so the generated numbers are completely wrong! It’s time to do damage control. You don’t care about order anymore; in fact, you’ll be satisfied as long as the list contains all of the numbers from 1 to **N**. Given the generated list, calulcate the number of <u>missing</u> integers.

INPUT

One line of space-separated integers. The first integer is **N**. **N** integers follow: the generated list.

CONSTRAINTS

$N < 10^6$

OUTPUT

Output the number of missing integers from the list.

SAMPLE TEST CASE 1:

INPUT:

```
10 1 2 5 9 11 3 3 2 6 -8
```

OUTPUT:

```
4
```

EXPLANATION:

The missing numbers from the list are 4, 7, 8, and 10.

4. Portals

PROBLEM

Mark is falling down the y-axis of a cartesian plane, and notices a bunch of one-way portals that teleport him from one x-value to another. Given the locations of all of these portals, give the ending x-position of Mark after falling through all of the portals. The portals themselves consist of two points sharing the same y-value. If Mark goes through the first point, he gets teleported to the second point.

No two portals will share a y-value. Additionally, the portals will be given in order of increasing y-value. As a result, the specific y-value of the portals are not important.

INPUT

The first line contains **N**, the number of portals.
The next **N** lines contains integers $\mathbf{x_1}$ $\mathbf{x_2}$, the two x-points that make up a portal.
The next line contains **T**, the number of positions to test.
The next **T** lines contain an integer **x**, a position for Mark to start out.

CONSTRAINTS

$0 \le \mathbf{N} \le 10^6$
$1 \le \mathbf{T} \le 10^5$
$-1000 < \mathbf{x}, \mathbf{x_1}, \mathbf{x_2} < 1000$

OUTPUT

The ending x-value of Mark after each position to test.

SAMPLE TEST CASE 1:
INPUT:

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

OUTPUT:

```
6
1
1
6
```

5. Math Instructions

PROBLEM

There is a sequence of operations that needs to be done on some integers! Given a list of instructions and then integers to do the operations on, give the results for each integer in the list. The possible instructions are (with division being integer division):

ADD <integer>
SUBTRACT <integer>
MULTIPLY BY <integer>
DIVIDE BY <integer>

INPUT

The first line contains **M**, the number of instructions. The next **M** lines contain the instructions. The next line contains **N**, the number of integers to operate on. The next **N** lines contain one integer **i**.

CONSTRAINTS

M < 100. There will be no division by zero in the instructions.
N < 1000

OUTPUT

Output the result of putting each integer **i** through the instructions.

SAMPLE TEST CASE 1:
INPUT:

```
5
ADD 5
MULTIPLY BY 10
SUBTRACT 8
DIVIDE BY 3
SUBTRACT -15
3
0
5
-6
```

OUTPUT:

```
29
45
9
```

6. Permutative yet Duplicative

PROBLEM

By the definition of a permutation, the number of permutations (ways to write) a string of length **n** is **n!**. However, in some cases, such as the string “aab”, some of the generated permutations will be identical! Given a string **S**, calculate the number of unique permutations of the string.

INPUT

One string **S**.

CONSTRAINTS

1 ≤ |**S**| ≤ 10. **S** only contains letters in the English alphabet.

OUTPUT

The number of duplicate permutations for that string.

SAMPLE TEST CASE 1:
INPUT:

```
aab
```

OUTPUT:

```
3
```

EXPLANATION:
The only permutations are “aab” “aba” and “baa”.

SAMPLE TEST CASE 2:
INPUT:

```
abcd
```

OUTPUT:

```
24
```

7. Ngon

PROBLEM

It's time to draw an Ngon! An Ngon is a polygon with **N** sides of equal length and equal angles between them. You'll be tasked with drawing an Ngon given the first two points of the polygon. You always draw the Ngon counter-clockwise. The first point acts as the starting point, and the second point is the next point you'd go to while drawing the Ngon. Given these two points, give the remaining points of the polygon.

INPUT

The first line contains **N**, the number of sides of the Ngon.
The next line contains $\mathbf{x_1}$ $\mathbf{y_1}$, the starting point.
The next line contains $\mathbf{x_2}$ $\mathbf{y_2}$, the second point.

CONSTRAINTS

$3 \le \mathbf{N} \le 50$
$-1000 \le \mathbf{x}, \mathbf{y} \le 1000$

OUTPUT

Output the remaining points, one point per line in the form **x y**.

SAMPLE TEST CASE 1:
INPUT:

```
4
0 5
0 0
```

OUTPUT:

```
5 0
5 5
```

SAMPLE TEST CASE 2:
INPUT:

```
3
-2 3
-23 24
```

OUTPUT:

```
-30.687 -4.687
```

8. Guess Counter

PROBLEM

This problem is so easy. You are given some sequence a_n, where $a_n = a_{n-1} + s$, where **s** is some constant, and a_0 is unknown. All you have to do is count the number of integers in a_n. Easy right? Oh, wait... I forgot to add SPACES between the integers in the input! I’m sure you can figure it out anyways.

INPUT

One string: a bunch of digits that make up the a_n sequence.

CONSTRAINTS

No integer in a_n will be less than or equal to zero nor contain more than 6 digits. The string of digits will always come from one and only one valid list. Additionally, 6 < |**input**| < 10000. Finally, a_n will have at least 3 items.

OUTPUT

The number of items in a_n

SAMPLE TEST CASE 1:

INPUT:

```
8910111213
```

OUTPUT:

```
6
```

EXPLANATION:

The list here is 8, 9, 10, 11, 12, 13.

SAMPLE TEST CASE 2:

INPUT:

```
89101113125
```

OUTPUT:

```
4
```

EXPLANATION:

The list here is 89, 101, 113, 125.

9. Chain Reaction

PROBLEM

Mark is making the best explosive display he's ever done in his life! To make it work, Mark has linked different fireworks together to the point that once one explodes, the other explodes after 5 seconds (goes both ways). Given Mark's links of fireworks, output how long Mark's explosive display will last, given that he lights the firework with index 0 first.

INPUT

The first line contains **N M**, the number of fireworks and the number of links, respectively.
The next **M** lines contain two integers **i j**, two fireworks that are linked together.

CONSTRAINTS

$0 < M < 10^5$
$0 \le$ **i, j** $<$ **N**
No fireworks will be double linked nor linked to themselves.

OUTPUT

Output the time at which the last firework(s) explodes.

SAMPLE TEST CASE 1:
INPUT:

```
30 7
0 1
1 2
2 3
1 3
3 4
2 4
0 2
```

OUTPUT:

```
10
```

EXPLANATION:
At time 0, firework 0 explodes, triggering fireworks 1 and 2.
At time 5, fireworks 1 and 2 explode, triggering 3 and 4.
At time 10, fireworks 3 and 4 explode.

10. Extreme Values

PROBLEM

Such an easy problem... all you have to do is multiply or add two numbers together! However, the values are so large they can break the the bounds of normal integers and longs. What will you do??

INPUT

The first line contains **N**, the number of test cases.
The next **N** lines begin with either “A” or “M” to show whether to add or multiply. This character will be followed by two integers which the operation should be conducted on.

CONSTRAINTS

N < 1000

OUTPUT

The value of the operation for each test case.

SAMPLE TEST CASE 1:
INPUT:

```
3
M 5126418764712685761872568124612 2859218590185819589218581 5215
A 5126418764712685761872568124612 2859218590185819589218581 5215
M
900049204910940129409191590195019250950912959250925920952195109019260
91029609609
182949817872171257918579187598175971759127587125981759817598175871285
791275981749872
```

OUTPUT:

```
146575518331439361681956993777264287805513952542580925571580
51550109506145439577647539398 27
164663838114449045209159612708150662104167180018428659160986334113543
354079459985940555710230243310683363802766957925048221141106957113918
04531176330141056845720048
```

NOTE:
The last 3 lines of the output make up only one line. In addition, the last 5 lines of the input make up only one line. Text wrapping makes the lines look like they are not one line.

11. Prefix Solver

PROBLEM

Given a prefix equation, solve it. Operators: + - * /

INPUT

One line of prefix, separated by spaces.

CONSTRAINTS

There will be no more than 1000 items in the list.

OUTPUT

The solution to the prefix equation.

SAMPLE TEST CASE 1:

INPUT:

```
+ - 5 3 2
```

OUTPUT:

```
4
```

SAMPLE TEST CASE 2:

INPUT:

```
/ * 6 + 1 1 – 4 + + + + 0 0 0 1 0
```

OUTPUT:

```
4
```

12. GCD and LCM

PROBLEM

Given 3 numbers, find the GCD and LCM of them.

INPUT

Three integers **a b c**

CONSTRAINTS

1 < **a, b, c** < 100

OUTPUT

Two lines: the GCD and then the LCM

SAMPLE TEST CASE 1:

INPUT:

```
10 5 15
```

OUTPUT:

```
5
30
```

SAMPLE TEST CASE 2:

INPUT:

```
7 11 13
```

OUTPUT:

```
1
1001
```

SAMPLE TEST CASE 3:

INPUT:

```
4 12 3
```

OUTPUT:

```
1
12
```

13. Spin Cycle

PROBLEM

To make a cool looking machine, Mark is aligning bicycle wheels like gears to where if he spins one, it also spins the wheels touching the ones Mark initially spun. Mark has two wheels already secured on a coordinate grid that are not touching. Mark also has **N** extra wheels that he also wants to place. Mark wants to place the wheels such all of them are ultimately connected. However, NONE of them can overlap; only their outer edge can touch. For each extra wheel, give a coordinate point they can be placed on to connect everything together with overlap!

INPUT

The first line contains the details of the first wheel: **x y r**
The second line contains the details of the second wheel: **x y r**
The third line contains **N**, the number of extra wheels. **N** is followed by **N** integers, each being the radii of the extra wheels.

CONSTRAINTS

-1000 < **x**, **y** < 1000
0 < **r** < 500

OUTPUT

Any list of coordinate points corresponding respectively to the list of radii. This should be checked to ensure it is correct. If the wheels cannot all be connected, output "NOT POSSIBLE".

SAMPLE TEST CASE 1:
INPUT:

```
0 0 6
0 20 18
1 4
```

OUTPUT:

```
0 7
```

14. Non-Random Explosion

PROBLEM

It's time to use random, but with a set seed **S**! You need to draw an explosion in a **N**x**N** grid (canvas). The explosion size should be proportional to **N**. Additionally, explosions with the same seed **S** should look near identical even if **N** is different. There is no sample output, as any good-looking explosion whose algorithm satisfies the random conditions will be accepted.

INPUT

On the first line, **N** is given.
On the second line, **S** is given.

CONSTRAINTS

30 < **N** < 500

OUTPUT

Output the NxN grid that makes up the explosion

SAMPLE TEST CASE 1:
INPUT:

```
40
125125923
```

15. Granite Rock

PROBLEM

We're digging through a mountain! Yep! It's not that bad, except for the fact that the rock seems to have a different density at different points at the mountain. The mountain can be represented by a 2D grid. We start at any point on the left and are trying to get to any point on the right. The 2D grid is a two-dimensional array of numbers, each representing the time it takes to dig through that unit of the grid. We don't need a straight path; only the fastest one. Assuming we can move up, down, left, or right, what is the shortest amount of time it would take to dig a path from left to right.

INPUT

The first line contains **R C**, the number of rows and columns in the mountain respectively.
The next **R** lines make up the mountain, each having **C** numeric characters from 0-9 representing the time needed to break through.

CONSTRAINTS

0 < **R**, **C** < 1000

OUTPUT

The shortest time needed to break a path through the mountain.

SAMPLE TEST CASE 1:

INPUT:

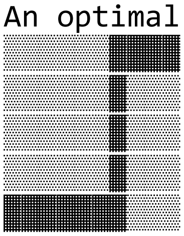
```
5 10
3933350003
0000331321
3310120031
3233332113
0000000334
```

OUTPUT:

```
6
```

EXPLANATION:

An optimal route:



16. Flip

PROBLEM

Mark needs to get some flippable integers, which are integers that can be reversed to get another integer. For example, 123 is a flippable integer as it can be reversed to get 321. 22 is also a flippable integer as it can be reversed to get 22. As a result, every integer is flippable!

What Mark forgot to realize is that he cannot accept every flippable integer. He has an inclusive range of integers that he must pick from. In addition, Mark will only accept an integer if both it and its flipped variant are both in the defined range. Given a range, find the number of flippable integers that satisfy this condition.

INPUT

Two space-separated integers **L** and **H**, the low and high bounds respectively.

CONSTRAINTS

$1 \leq \mathbf{L} \leq \mathbf{H} \leq 10^{20}$

OUTPUT

Output the number of flippable integers in range.

SAMPLE TEST CASE 1:

INPUT:

```
1 100
```

OUTPUT:

```
100
```

SAMPLE TEST CASE 2:

INPUT:

```
13 25
```

OUTPUT:

```
1
```

17. Tree Dating!

PROBLEM

After too many negative dating experiences, Mark gave up on humans and decided to take on his newest passion: trees. However, Mark has very high standards, and only will accept trees that are symmetrical, meaning that they can be flipped horizontally and still produce the same image. Each tree starts at one point (the root) and splits up into some number of branches, from a list. Every branch repeats this process until the $\mathbf{D}^{th}$ branch is reached. Given this, how many unique symmetrical trees given the constraints?

INPUT

The first line contains **N**, the number of integers on the next line.
The next line contains **N** integers $\mathbf{n_i}$, all of which a possible number of branches a node could split into.
The next line contains **D**, the height of the tree.

CONSTRAINTS

$0 < \mathbf{N} < 13$
$0 < \mathbf{n_i} < 13$
$0 < \mathbf{D} < 1013$

OUTPUT

Output the total number of possible symmetrical trees, modulo $10^9 + 7$.

SAMPLE TEST CASE 1:

INPUT:

```
3
1 2 3
2
```

OUTPUT:

```
15
```

EXPLANATION:

The trees:

1->1 1->2 1->3 2->1,1 2->2,2 2->3,3 3->1,1,1 3->1,2,1 3->1,3,1
3->2,1,2 3->2,2,2 3->2,3,2 3->3,1,3 3->3,2,3 3->3,3,3

18. Walk in the Park

PROBLEM

Mark is planning the perfect outdoor walk in his favorite local park. He is bringing a good friend and wants to make sure this is a walk to remember. The park can represent a graph consisting of a bunch of paths that interest at points. Each path has a “beauty value,” a positive integer representation of the beauty of that path. Mark would walk every path if he could, but he doesn’t want to make the walk too long and is limiting himself to at most **P** paths. Additionally, he wants to always end up back at his starting point (node/intersection 0) and does not want to repeat any paths on his walk. Given these constraints, what is the highest total “beauty points” he could achieve on one walk.

INPUT

The first line contains **N M P**, the number of intersections (nodes), (edges), and the path limit respectively.
The next **M** lines contain the values **i j b**, where **i** and **j** are the two nodes connecting the path and **b** is the beauty value of that path.

CONSTRAINTS

$2 < \mathbf{P} \le \mathbf{N} < 1000$
$0 < \mathbf{M} < 1000$
There exists at least one possible loop that Mark can walk on. No two nodes will have multiple paths between them.

OUTPUT

The highest total “beauty points” achievable.

SAMPLE TEST CASE 1:
INPUT:

```
5 7 3
0 1 3
1 2 1
2 3 1
0 2 5
0 4 3
1 4 3
0 3 4
```

OUTPUT:

```
10
```

EXPLANATION:
The path: 0 -> 2 -> 3 -> 0

19. Magnetic Field

PROBLEM

I own a magnetic bar that rests on a coordinate plane, taking up the area from x = 0 to x = 1000. Additionally, there also rest magnetic balls at x-locations that exert a force on the bar. The force exerted on this bar from each ball is not evenly distributed; the force exerted at some x-location on the bar is equal to max(0, **W** - **s**|**d**|), where **W** and **s** are constants unique to each ball and **d** is the distance from the ball to a certain x-value on the bar. At what x-value is the force the largest?

INPUT

The first line contains **N**, the number of balls.
The next **N** lines contain the values **x W**, the x value of a ball and its **W** value respectively.

CONSTRAINTS

1 < **N** < 10000
-500 < **x** < 1500
0 < **W** < 1500
1 ≤ **s** < 100

OUTPUT

The x-value on the bar where the force is the greatest.

SAMPLE TEST CASE 1:
INPUT:

```
3
500 1000 100
502 1000 200
506 1000 200
```

OUTPUT:

```
502
```

20. Radiant Pearl

PROBLEM

You're searching for the rarest pearl in the ocean: a pearl that gives off so much energy its location can be somewhat tracked using a tracking device. This pearl exists at somepoint along the ocean floor, and can move right or left at a rate of **S** meters per hour. Each time you dip your tracking device into the water, it gives back a reading of a double value where $R = \frac{1}{kd}$, where **R** is the reading, **k** is some constant, and **d** is the distance from your tracking device to the pearl.

To solve this problem, you must setup a multithreaded program that communicate through a stream. One program, launched from the main, will read in the input. This program will also give you the value of **R** each time you call it by sending in the location of your tracking device. The original program (main thread) must call use these values of **R** to eventually decode the location of the pearl.

You only have 3 hours! Additionally, you must spend one hour between each reading (it takes time to analyze data and move your boat after all). That means you have to know the location of the pearl after 3 readings.

INPUT

One line containing **X S k**, the initial x-location of the pearl, the speed of the pearl, and a constant respectively.

CONSTRAINTS

-1000 < **X** < 1000
-100 < **S** < 100
0.25 < **k** < 12.5

OUTPUT

If you send in the location of the pearl ($\pm 10^{-2}$) at or before the 3 hour mark, the program outputs "PEARL FOUND!" If you pass 3 hours without the pearl, the program outputs "FAILED!"

SAMPLE TEST CASE 1:

INPUT:

```
-200 10 1
```

OUTPUT:

```
PEARL FOUND!
```