7^{th} Virginia Tech High School Programming Contest Division I

Dec 12, 2020

As a reminder, here are the key rules under which this contest is conducted:

- Contestants may not communicate with another human during the contest about the problems.
- This is an individual (not a team) contest.

This problem set is targeted at intermediate and experienced contestants, those only starting to participate in programming contests should consider Division II.

Enjoy!

This page is intentionally left blank.

Problem A
Colorland


sample003.in

Yancy is designing a new board game called Colorland. The rules of Colorland are as follows:

- Colorland's board is a linear sequence of $N + 1$ squares. The first square is a special start square, the remaining N squares are colored blue, orange, pink, green, red, or yellow.
- The goal of Colorland is to move your game piece from the start square to square N.
- Players take turns drawing from a shuffled deck of cards. Each card has a single color on it. A player moves to the nearest square after their current square with the color indicated by the drawn card.
- If there are no squares of the drawn color after a player's current square, that player's piece does not move and their turn ends.

Yancy is interested in the length of play required for different board layouts. She would like to know the smallest number of cards any one player would have to draw to complete the game.

For instance, the board for sample 3 is [`Start`, `Blue`, `Orange`, `Pink`, `Green`, `Red`, `Yellow`, `Yellow`, `Yellow`, `Yellow`]. The best first draw is `Yellow` which advances a player from Start to the 6^{th} square. From the 6^{th} square to the end, only a `Yellow` draw will advance the player. Therefore the smallest number of draws is 4.

Input

The first line of input consists of a single integer N ($1 \leq N \leq 200\,000$) denoting the number of squares. The next N lines each contain a single string $S_i \in \{Blue, Orange, Pink, Green, Red, Yellow\}$ representing a color of the i^{th} square, starting with the 1^{st} square on the board.

Output

A single line containing a single integer equal to the minimum number of draws required to move from the start square to square N.

Sample Input 1	Sample Output 1
6 Blue Orange Pink Green Red Yellow	1

Sample Input 2

```
12
Blue
Orange
Pink
Green
Red
Yellow
Yellow
Red
Green
Pink
Orange
Blue
```

Sample Output 2

```
2
```

Sample Input 3

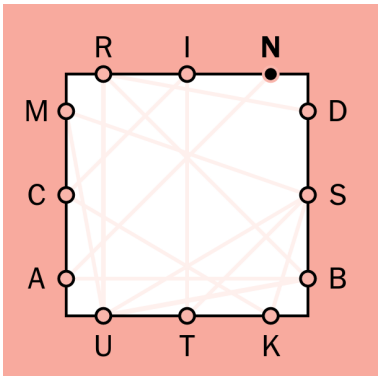
```
9
Blue
Orange
Pink
Green
Red
Yellow
Yellow
Yellow
Yellow
```

Sample Output 3

```
4
```

Problem B
Boxed Letters

The New York Times publishes a puzzle called "Boxed Letters" for puzzle enthusiasts. In this puzzle, 12 letters are shown along the 4 sides of a box. The task is to find words that are spelled when these letters are connected, subject to the following rules:



An example of the NY Times "Boxed Letters" Puzzle. This sample corresponds to sample input 1.

- You may start the first word with any of the letters
- Consecutive letters in one word cannot be from the same side
- Words must be at least 3 letters long
- Letters can be reused
- The last letter of a word becomes the first letter of the next word, e.g. `THY > YES > SINCE`
- All letters must be used
- Use as few words as possible

After trying the puzzle for a while, you remember that there are dictionaries that should make this task simple.

Can you write a program that solves a Boxed Letters puzzle given a dictionary of words?

Input

The input consists of a single test case, which starts with an integer n ($1 \leq n \leq 75\,000$), the size of the dictionary. This line is followed by n lines, each containing a dictionary word of l uppercase English letters ($3 \leq l \leq 22$). All dictionary words are distinct. The next 5 lines contains the letters around the box in compressed form. The first line contains the top row, the last line the bottom row. The first character on each of the 3 lines in between corresponds to the box's left side, the second character corresponds to the box's right side. All 12 characters are distinct uppercase English letters.

Output

Output a word list that solves this puzzle, in the order in which the words would be entered into the puzzle. Place each word on a separate line. If there are multiple word lists that satisfy the conditions, you may output any of them. You are guaranteed that a solution exists.

Sample Input 1	Sample Output 1
22 DATUM TRUMAN STAINS INSTANTS TACITUS LINUS TANTRUMS MCADAM ILLICIT ALUM CARINA DRUMSTICKS BACILLI SUBURBAN TIC SALSA TALIBAN ATTAIN LUNATIC ASIATICS MANDARIN RUMINANTS R I N M D C S A B U T K	DRUMSTICKS SUBURBAN

Problem C
Cave Exploration

Source: Photo by macg on pixabay

It is monsoon season, and your goldfish Orange is stuck at the bottom of a cave system in Thailand. Every hour, the water rises by 1 meter. Thankfully, Orange has the ability to swim instaneously from one location to another. However, he can't hop over the sharp rocks in the way.

You trust that Orange is smart and will find the way out of the cave as soon as possible. To prepare Orange's meal, you want to write a program to find out how long it will take for Orange to come back home.

You are given the cave system as an $N \times N$ grid where each location (i, j) contains a nonnegative integer, $h_{i,j}$, representing the height of the rocks in meters at that location. Orange can only swim left, right, up, or down from one submerged location to another submerged location (but not diagonally).

A location (i, j) is submerged if the water level is at least 1 meter higher than $h_{i,j}$. Orange's starting location will always be at $(0, 0)$, with $h_{0,0} = 0$. The initial water level is 1 meter so that Orange's start location is submerged. The only exit in the cave is at location $(N - 1, N - 1)$. After how many hours can Orange find his way to the exit?

Input

The input consists of one integer, N, on the first line such that $2 \leq N \leq 100$, followed by N lines with N integers each, separated by spaces, such that each integer represents the height $h_{i,j}$ ($1 \leq h_{i,j} \leq 10^8$) of the rocks at location (i, j).

Output

Output a single integer, which is the minimum number of hours that must pass before Orange can reach the exit.

Sample Input 1

```
2
0 3
2 4
```

Sample Output 1

```
4
```

Sample Input 2

```
5
0 2 3 10 4
10 23 2 21 12
11 10 12 12 16
12 12 18 10 10
10 10 10 11 10
```

Sample Output 2

```
12
```

This page is intentionally left blank.

Problem D
Cafeteria

Theta likes to play Lure of the Labyrinth, which is an online game that uses a compelling graphic novel storyline to engage middle grades students in mathematical thinking and problem-solving. To find lost pets, students have to infiltrate a world of monsters and solve puzzles! Implemented by a professional game studio, these puzzles are quite engaging and challenging.

Source: Lure of the Labyrinth, used with permission of Maryland Public Television (MPT)

In the manager's cafeteria, students are asked to practice proportions by serving 5 monsters sitting at the table: Salamander, Yeti, Golem, Imp, and Kraken. Each monster wants some amount of burger, slob, sushi, and drumstick.

The amount of each item each monster wants on their plate is proportional to the amount of each of the other items that monster wants, and the proportionality ratio is the same for all monsters.

For instance, as shown in sample input 1 and in the accompanying figure, if Golem (center) wants 6 amount of burger on his plate and 12 amount of drumstick, and Salamander (left) wants 40 amount of drumstick, then Salamander will want 20 amount of burger. Students practicing proportions will program the dispenser machine to release 20 amount of burger. Similary, Kraken (right) wants 36 amount of drumstick because it has 81 amount of slob on its plate and Salamander has 40 amount of drumstick and 90 amount of slob. If the students compute all proportions in time, the monsters eat and a pet can be saved!

As students progress in the game and reach more difficult levels, fewer and fewer amounts are given, requiring more intuition and thinking to solve the puzzle.

Give a set of partially filled plates, write a program that computes the number of distinct solutions that are consistent with it!

Input

The input consists of 2 lines of 10 entries each describing the partially filled plates. The first line describes the top row (burgers and slob), the second line describes the bottom row (sushi and drumstick). On each line, the first 2 entries describe Salamander's plate, the next 2 Yeti's, then Golem's, Imp's, and finally Kraken's. Each entry is either the underscore character _ describing an empty slot or a positive integer number a ($0 < a \leq 200$) if it is already known. Entries are separated by single spaces. You may assume that each arrangement has at least one possible solution (that is, the partial information is not inconsistent).

Output

Output the number n of distinct solutions for the puzzle given in the input! If there are infinitely many solutions, output "`many`"! Note that although any given entries are guaranteed to be less than 200, inferred entries must be positive integers, but they are not subject to a maximum.

Sample Input 1

```
_ 90 22 _ 6 _ _ _ _ 81
_ 40 _ _ _ 12 60 _ 90 _
```

Sample Output 1

```
1
```

Sample Input 2

```
85 55 _ 99 51 _ _ _ _ _
_ _ _ _ _ _ _ 85 63 153
```

Sample Output 2

```
1
```

Sample Input 3

```
160 _ _ 136 _ _ _ _ _ 170
_ _ _ _ 120 _ _ 144 _ _
```

Sample Output 3

```
8640
```

Sample Input 4

```
36 99 _ 55 _ 99 _ 77 _ _
_ 144 _ _ 27 _ 21 112 _ _
```

Sample Output 4

```
many
```

Problem E
Ball Colors


Image by Gerhard Bögner from Pixabay

Baby Timmy has a pool of balls that he plays with. The pool is made up of balls with n different colors with a certain number of each color. Baby Timmy finds it interesting to order these balls in a line given certain conditions he made. Timmy has two conditions when playing his own game:

- Balls of certain colors may not be next to each other
- One particular sequence of colors which Timmy likes most must appear as often as possible in the arrangement

Can you compute the total number of different ways in which Timmy can arrange his balls?

For instance, suppose Timmy has 2 red, 1 yellow, 2 green, and 1 blue ball(s). He doesn't like for red or yellow balls to be next to each other, and his favorite sequence is "green blue." The following six arrangements meet the requirements:

```
red green blue red green yellow
red green blue yellow green red
red green red green blue yellow
red green yellow green blue red
yellow green blue red green red
yellow green red green blue red
```

This arrangement corresponds to Sample Input 1.

Input

The input consists of a single test case with three lines. The first line contains an integer n ($2 \leq n \leq 50$), which describe the number of different colors. The remaining n integer on that line denote how many balls Timmy has of each color (colors are numbered 1 through n and their frequencies appear in order).

The second line of input describes which colors Timmy does not want next to each other. The first integer k ($0 \leq k \leq n$) gives the number of colors. This is followed by k integers c_i ($1 \leq c_i \leq n$) denoting the colors that cannot be next to each other. Each c_i is unique.

The third line of input describes the sequence Timmy likes most. This first integer l ($0 \leq l \leq n$) describes the length of this sequence, and the following l integers s_i ($1 \leq s_i \leq n$) describe the sequence that must appear as often as possible in the arrangement. Each s_i is unique and the sets $\{c_i\}$ and $\{s_i\}$ do not intersect.

Output

Output the number of arrangements Timmy can make that satisfy the given conditions. Since the number can be large, output its value modulo $1\,000\,000\,007$.

Sample Input 1

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

Sample Output 1

```
6
```

Sample Input 2

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

Sample Output 2

```
18
```

Sample Input 3

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

Sample Output 3

```
12
```

Sample Input 4

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

Sample Output 4

```
0
```

Problem F
Faulty Sprinklers

Source: Flicker/sprinkler.org

Bob's green thumb has recently gotten him a job as a gardener at a local park. Along with caring for all the plants in the garden, he was also given the responsibility of completely redesigning the park with all new equipment and green spaces.

Unfortunately for Bob, he's not as good at financing as he is at gardening, and may have messed up. In one area of the garden, he has a courtyard in the shape of a perfect square. In the process of designing the courtyard, he spent so much money on fertilizer and grass seed that he had to buy cheap sprinklers.

When the time came to install the sprinklers, Bob found out that the sprinklers don't all rotate the same amount, and not all of the grass is getting water. Assuming that each spinkler can spray water infinitely far, help Bob determine the proportion of the courtyard that is watered by the sprinklers.

Input

The input consists of a single line containing four real numbers a, b, c, and d ($0 \leq a, b, c, d \leq 90$) representing the sizes of four angles in degrees. The number a represents the angle that the bottom right sprinkler rotates from the right wall; similarly, b is the angle that the top right sprinkler rotates from the top wall, c is the angle that the top left sprinkler rotates from the left wall, and d is the angle that the bottom left sprinkler rotates from the bottom wall. All numbers will be given with no more than 3 decimals after the decimal point.

Output

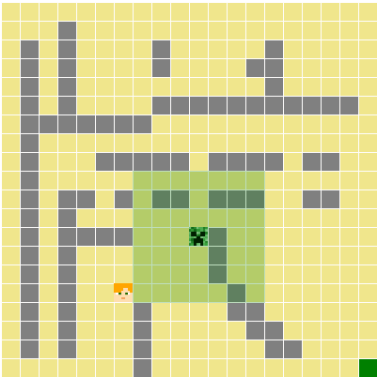
Output a single double, the proportion of the area of the courtyard that is watered by the sprinklers. Your answer will be considerd correct if it is within 10^{-6} of the correct answer.

Sample Input 1	Sample Output 1
45 45 0 0	0.75

Sample Input 2	Sample Output 2
30 30 10 45	0.87044439473

This page is intentionally left blank.

Problem G
Minecraft Dungeons



Sample Input 1

During the COVID-19 quarantine, Theta discovered Minecraft Dungeons which is an offshoot of the popular Minecraft game. In Minecraft Dungeons, players crawl through a dungeon, trying to accomplish a mission without getting killed by various mobs. At the end, a boss battle with the Arch-Illager awaits.

Fortunately, the mobs' AI isn't particularly strong, so with some planning, it's possible to avoid the mobs. In this problem, you're given a dungeon map and you need to find out if it's possible for the player to reach the exit without being blown up by a creeper.

You, the player, can move north, south, east, and west, except where there are obstacles. The player may also stay put. There is one creeper on the map. The player and the creeper take turns simultaneously, subject to the following simple AI:

- The creeper will only try to move towards the player, either horizontally or vertically, in a way that reduces the horizontal or vertical distance.
- If an obstacle prevents the creeper from moving vertically, it will try to move horizontally, and vice versa.
- If it can't move in either direction, it will stay put.
- If the creeper can move both horizontally and vertically, it will choose the direction in which it is farther away from the player. In the case of a tie, it will move vertically towards the player.

After each such turn, the game checks if the creeper is too close to the player, in which case the creeper will explode and kill the player, even if the player has already reached the exit in that move. Obstacles do not protect the player from explosions. If the player didn't explode and reached the exit the mission is completed.

Input

The input consists of a single test case. The first line contains three integers n ($1 \leq n \leq 30$), m ($1 \leq m \leq 30$), and e ($1 \leq e \leq \min(n, m)$). e specifies the creeper's explosion "radius" - the creeper will explode if both the horizontal and the vertical distance between player and creeper is less than or equal to e.

The following n lines consist of m characters each and describe the dungeon map using the following characters

- `P` - the start position of the player
- `C` - the start position of the creeper
- `E` - the position of the exit

- `X` - an obstacle
- `.` - an empty square that can be entered by both creeper and player

There is exactly one '`E`', '`P`', and '`C`' each in the input.

Output

If it is possible to complete the mission and reach the exit without being blown up by the creeper, output the minimum number of moves to escape. A move consists of either the player or the creeper moving, or both. If it is not possible to escape, print "`you're toast`"!

Sample Input 1

```
20 20 3
....................
...X................
.X.X....X.....X.....
.X.X....X....XX.....
.X.X..........X.....
.X.X....XXXXXXXXXX.
.XXXXXX.............
.X..................
.X...XXXXX.XXXX.XX..
.X..P............C..
.X.XX.X.XX.XXX..XX..
.X.X................
.X.XXXX....X........
.X.X.......X........
.X.X.......X........
.X.X........X.......
.X.X...X....XX......
.X.X...X.....XX.....
.X.X...X......XX....
.......X...........E
```

Sample Output 1

```
119
```

Sample Input 2

```
5 5 1
E...C
.....
XXXX.
.....
P....
```

Sample Output 2

```
you're toast
```

Sample Input 3	Sample Output 3
5 5 3 E.... ...XX ...XC ...XX P....	4

This page is intentionally left blank.

Problem H
Powers of 2

Theta has been learning about powers of 2 in school. She notices that some numbers when written out contain powers of 2 in their digit representation: for instance, 12 560 contains 256 which is a power of 2. She has been wondering how many such numbers there are.

Can you write a program that counts how many numbers contain a given power of 2?

Powers of 2

Input

The input consists of a single line with two integers n and e ($0 \leq n \leq 9 \cdot 10^{18}, 0 \leq e \leq 62$).

Output

Output a single integer that is equal to the number of distinct integers k ($0 \leq k \leq n$) whose decimal representation contains the digits of 2^e as a substring.

Sample Input 1	Sample Output 1
1000000 1	468559

Sample Input 2	Sample Output 2
1000000 5	49401

Sample Input 3	Sample Output 3
1000000 16	20

Sample Input 4	Sample Output 4
9000000000000000000 62	1

Sample Input 5	Sample Output 5
5432123456789876543 33	4842258985

This page is intentionally left blank.

Problem I
Paludarium

Source: Photo by Sventie

Feeling lonely in his apartment since he is quarantined inside at the moment, Bob decided to make use of the fish tank left by the previous renter to set up a little paludarium (semi-aquatic habitat that combine land and water environment) and fill it with little creatures to keep him company at home.

As he is testing out different layouts, Bob wonders if his landscape would provide a good ratio of water and air space for his new pets. To simplify this problem, Bob imagines the tank to be a 2D grid which will be filled with solid land blocks of different heights from his current landscape construction. To achieve the optimal balance, his goal is to figure out what is the minimum difference he can achieve between the number of square blocks occupied with water versus the number of square blocks that are occupied with air in his tank. As Bob fills water into the tank, the water will rise evenly across an entire block layer at a time, so any blocks of air at that level will be replaced with blocks of water. Each square is either water, air, or land (fractional squares are not possible in Bob's design).

Given a list of integers representing the different land block heights present in the tank from left to right, output the water level necessary to minimize the difference between water and air to achieve optimal balance. If there are multiple water levels that can achieve the optimal balance, output the smallest such level. Also compute the amount of water that will be needed.

For example, suppose the layout of the tank can be represented by a list of integers $[0, 0, 2, 0, 1]$ where each integer is the height of land blocks at a column in the tank, counted from left to right, with a tank's overall height of 3 and width of 5 (as shown by the illustration below which corresponds to Sample Input 1).

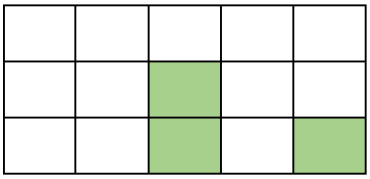


Figure I.1: Tank layout

As shown in Figure I.2, filling up to a height of 1 with water results in 3 water blocks and 9 air blocks (a difference of 6). As shown in Figure I.3, filling up to a height of 2 with water results in 7 water blocks and

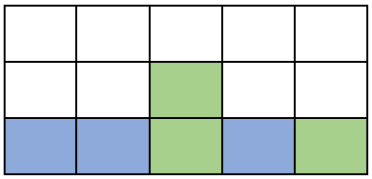


Figure I.2: Tank layout with water level of 1

5 air blocks (for a difference of 2), which is the optimal difference that Bob wants for this input.

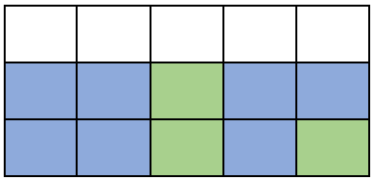


Figure I.3: Tank layout with water level of 2

Input

The input consists of 2 lines. The first line contains two integers H and W describing the height and width of the tank ($1 \leq H \leq 10^9$, $1 \leq W \leq 5 * 10^5$). The second line contains W integers B_i where each B_i is the height of the land blocks at the i^{th} column of the tank, counted from left to right ($0 \leq B_i \leq H$ for $i = 1 \ldots W$).

Output

Print two integers, the first one describing the smallest water level that achieves the optimal balance and the second one providing the total number of water blocks in Bob's tank at this level. If adding no water yields the optimal balance, output `0 0`.

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

Sample Input 2	Sample Output 2
6 6 4 0 3 5 2 1	4 10