Rosswell wolves and cows **P69326_en**

Vintè Concurs de Programació de la UPC - Semifinal (2021-06-15)

It is unfortunate that, due to the messy state of the world we live in, the publication of the memories of Lieutenant Coronel Sargent Major John R. Smith III, Phd, MD, MotU by the United States Air Force has remained mostly unnoticed by the public.

Officer Smith was in charge of the Roswell Army Air Field during the 1940s, and in his memories he finally confirms what conspiracy theorists have suspected during the last decades: the Roswell incident was indeed an extra-terrestrial encounter—albeit of a different kind than expected.

According to Smith, after following the trajectory of a falling object on the sky, soldiers of the United States Air Force encountered the debris of a space ship. The search for the pilot ended in the most confusing way: the cows that grazed in the area had instinctively started chasing and corralling the confused alien, which turned out to be a wolf-like creature, across the checkered corn and soy fields of New Mexico. The traveller was completely surrounded by the animals and could not escape when the soldiers arrived.

In retrospect, Smith wonders if a better choice for a landing site would have given the alien wolf a chance to escape. He then enters a long digression about hypotheticals, possibilities, alternate realities, and in particular one where he has been given the medal of military valour which he clearly deserved for his continued service but which *"those [edited] burocrats from DC never thought of"*.

Input

Input consists of several cases, each one starting with the dimensions H and W of the area around Roswell, which can be represented as a chess board. Follow H lines, each containing W characters. A dot corresponds to an empty position, whereas '`C`' corresponds to a cow. All cows will be in the white squares of the board, with the top left one being white. Assume that H and W are between 1 and 32, and that there are at most 12 cows on each board.

Cows and the wolf move by turns, never leaving the board. In the first turn, the alien wolf can land in any white square which is not occupied by a cow. There will always be at least one such square.

Afterwards, in each turn for the cows, one of them must move to a diagonally adjacent square of the row above their current position which is not occupied by the wolf or another cow. If no cow can move, their turn is skipped and the wolf moves again.

In subsequent turns for the wolf, it must move to one of the four diagonally adjacent squares that is not occupied by a cow. If all four positions are occupied, the wolf has been captured. The wolf succeeds to escape if it reaches a square in the bottom row.

Output

For each case, print the number of landing squares for the alien wolf from where there exists a winning sequence for it. Do not assume any strategy by neither wolf nor cows.

Sample input

```
1 1
.

4 4
....
....
....
.C.C

4 4
....
...C
C.C.
.C.C

6 8
........
........
........
........
........
.C.C.C.C

2 3
...
...

3 4
....
.C..
..C.

2 3
C..
.C.

3 4
....
...C
C.C.
```

Sample output

```
1
6
0
20
3
4
0
2
```

Problem information

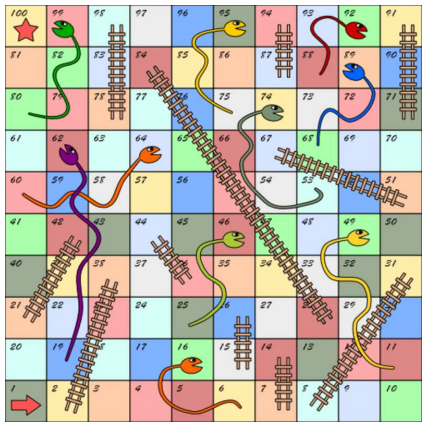
Author : Edgar Gonzàlez
Generation : 2022-06-13 21:22:20

Lazy Snakes and Ladders (2) — P65875_en

Vintè Concurs de Programació de la UPC - Semifinal (2022-06-15)

The author of the previous problem is half a psycho, half a freak. He is well know for his lengthy and weird statements, although *a few* of his problems are not that hard. Consider for instance the following adaptation of an old problem of the same author.

The rules of *Snakes and Ladders* (see the board below) are simple:



- Players red and blue start with their counters on cell number 1, and take turns in rolling a six-sided die, with red going first.
- The counter for the current player moves forward the number of cells rolled in the die (e.g., rolling a 5 when on cell 4 takes the counter to 9).
- The goal is to reach the cell 100. An exact roll is needed: in case of excess, the counter *bounces* and moves the extra count backwards (e.g., rolling a 5 when on cell 97 takes the counter to 98).
- If the landing cell (after potential bouncing) is the bottom of a ladder, the counter is moved to its top, which will be a higher-numbered cell (e.g., rolling a 1 when on cell 1 takes the counter to 38). Nothing happens when the counter directly lands on a top.
- If the landing cell (after potential bouncing) is the head of a snake, the counter is moved to its tail, which will be a lower-numbered one (e.g., rolling a 3 when on cell 98 takes the counter to 80). Nothing happens when the counter directly lands on a tail.
- If the rolled number was six, the player keeps the turn; otherwise, it passes to the other player (irrespective of whether bouncing, snakes, or ladders were involved).

You must simulate several of these games using pseudo-random numbers. In particular, include the `<random>` library, and declare a global

```
mt19937 rng;
```

variable. Every game will be defined by a seed s. Just do

```
rng.seed(s);
```

to reset `rng` before every game. Afterwards, every time that you need the result d of the next rolling of the die, use this code:

```
unsigned int r = rng();
int d = r%6 + 1;
```

For instance, with the initial seed 42, we get these values for r: 1608637542, 3421126067, 4083286876, 787846414, ... Therefore, the values for d are 1, 6, 5, 5, ... In this game, red goes to 2 (and then to 38), blue goes to 7 (and then to 14), blue moves again (he got a 6) and goes to 19, red goes to 43, ..., and eventually blue wins.

Input

Input consists of several games, each one defined by an integer seed s between 1 and 10^9.

Output

For each game, print "`RED`" or "`BLUE`" depending on the winner.

Sample input	Sample output

`42`
`1`
`999999999`
`1000000000`
	`BLUE`
	`BLUE`
	`RED`
	`BLUE`

Problem information

Author : Edgar Gonzàlez
Generation : 2022-06-13 11:40:00

Optimal blue-red tree **P24951_en**

Vintè Concurs de Programació de la UPC - Semifinal (2021-06-15)

You are given an undirected connected graph with no cycles. You must paint every node either blue or red. Painting in blue costs 1 per node, while painting in red costs 2 per node. Your goal is to minimize the total cost of painting the tree. There is just one restriction: Each node can have, at most, one adjacent node with the same color than itself.

Input

Input consists of several trees, each one with the number of nodes n, followed by $n-1$ pairs x y for the edges. Nodes are numbered from 0. Assume $1 \leq n \leq 10^5$.

Output

Print the minimum cost to color each tree.

Sample input

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

Sample output

```
1
4
6
10
```

Problem information

Author : Salvador Roura
Generation : 2022-06-13 21:16:04

Average and median **P67363_en**

Vintè Concurs de Programació de la UPC - Semifinal (2021-06-15)

Consider a set $S = \{x_1, \ldots, x_k\}$ of natural numbers (maybe with repetitions), with odd k. The *average* of S is defined as $(\sum_{1 \leq i \leq k} x_i)/k$. The *median* of S is defined as the element that is in the middle of the set after we sort it. For instance, for $S = \{1, 2, 2, 4, 5\}$, the average is $(1 + 2 + 2 + 4 + 5)/5 = 14/5 = 2.8$, and the median is 2.

You are given a set of n natural numbers, with even n. Remove exactly one element so as to maximize the absolute value of the difference between the average and the median.

Input

Input consists of several cases, each one with an even n, followed by n natural numbers between 0 and 10^9. Assume $4 \leq n \leq 10^5$.

Output

Print the maxim possible difference between the average and the median, with two digits after the decimal point. To do so, include these two lines at the beginning of your main:

```
cout.setf(ios::fixed);
cout.precision(2);
```

The input cases do not have precision issues.

Sample input

```
6  1 2 2 3 4 5
8  2 1 4 8 0 3 9 2
4  999999999 1000000000 999999998 999999998
4  0 1 2 9
4  0 7 8 9
```

Sample output

```
0.80
1.71
0.67
2.33
2.33
```

Problem information

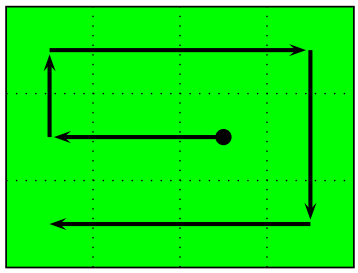
Author : Félix Moreno
Generation : 2022-06-11 13:52:48

Milan cemetery **P85166_en**

Vintè Concurs de Programació de la UPC - Semifinal (2021-06-15)

Legend has it that SWERC teams have a curse with cemeteries. This year, Xavier and Izan woke up the day before the competition in Milan's cemetery. It was so monumental that they decided to visit everything. However, they didn't want to get too tired.

The cemetery is an $n \times m$ grid, and they woke up in position (x, y). Can you help them figure out the minimum number of changes of direction needed to visit every cell? They can only move in the N, S, E, O directions.



For instance, to the right you can see the first case of the sample. It is possible to visit all the cells with only four changes of direction.

Input

Input consists of several cases, each one with n, m, x and y. Assume that both n and m are between 1 and 10^6, $1 \leq x \leq n$, and $1 \leq y \leq m$.

Output

For each case, print the minimum number of changes of direction required to visit the whole cemetery.

Sample input

```
3 4 2 3
1 1 1 1
9 1 5 1
```

Sample output

```
4
0
1
```

Problem information

Author : Edgar Moreno
Generation : 2022-06-13 21:24:02

Some parenthesizations **P31589_en**

Vintè Concurs de Programació de la UPC - Semifinal (2021-06-15)

Given a natural number n, compute the number of correct parenthesizations with n opening and n closing parentheses, with just one restriction: the substring "`)()`" is not allowed.

For instance, for $n = 4$ there are four correct parenthesizations: "`(((())))`", "`(()(()))`", "`(())(())`" and "`()((()))`".

Input

Input consists of several cases, each one with an n between 1 and 10^4.

Output

For each n, print the result modulo $10^9 + 7$.

Sample input

```
4
1
2
3
5
10000
```

Sample output

```
4
1
1
2
9
681928184
```

Problem information

Author : Xavier Povill
Generation : 2022-06-11 11:20:48

Arithmetic derivative **P86263_en**

Vintè Concurs de Programació de la UPC - Semifinal (2021-06-15)

Given a natural number n, its arithmetic derivative $d(n)$ is defined as follows:

- $d(0) = d(1) = 0$.
- If n is prime, then $d(n) = 1$.
- Let $n = x \cdot y$, with $1 < x, y < n$. Then $d(n) = x \cdot d(y) + y \cdot d(x)$.

For instance, $d(4) = 2d(2) + 2d(2) = 2 + 2 = 4$, and $d(6) = 3d(2) + 2d(3) = 3 + 2 = 5$. It can be proven that this definition is consistent. For example, $d(12) = 4d(3) + 3d(4) = 4 + 12 = 16$, and also $d(12) = 6d(2) + 2d(6) = 6 + 10 = 16$.

We say that f is a fixed point of $d(n)$ if $d(f) = f$. For instance, 0 and 4 are fixed points. Given ℓ and r, can you compute the number of fixed points of $d(n)$ in $[\ell..r]$?

Input

Input consists of several cases, each one with ℓ and r, with $0 \leq \ell \leq r \leq 10^{18}$.

Output

For each case, print the number of fixed points of $d(n)$ in $[\ell..r]$.

Sample input	Sample output
0 4	2
1 20	1
4 4	1
5 23	0
900000000000000000 1000000000000000000	0

Problem information

Author : Salvador Roura
Generation : 2022-06-11 19:11:17

Counting suffixes **P88868_en**

Vintè Concurs de Programació de la UPC - Semifinal (2021-06-15)

Please implement a data structure D to efficiently support four operations:

- `I` s: Inserts the string s into D. No changes are made if s is already in D.
- `E` s: Erases the string s from D. No changes are made if s is not in D.
- `C` s: Counts the number of strings in D that end with the suffix s.
- `R`: Resets D, that is, removes all strings from D.

Input

Input consists of several operations over an initially empty D. Assume that each s is made up of between 1 and 100 lowercase letters. At no moment the sum of the sizes of the strings stored in D will be larger than 10^6.

Output

Print the result of each `C` s operation, and three dashes for each `R` operation.

Sample input

```
E a
I abba
C a
I cba
C cba
C abba
C a
C ba
I abba
C ba
E cba
C cba
C a
E ba
C ba
R
C ba
I eggs
I zzeggs
C eggs
E eggs
C eggs
```

Sample output

```
1
1
1
2
2
2
0
1
1
---
0
2
1
```

Problem information

Author : Salvador Roura
Generation : 2022-06-13 14:57:04

No Brainer **P76081_en**

Vintè Concurs de Programació de la UPC - Semifinal (2021-06-15)

The UPC programming competition is twenty years old! This implies that many of the participants of today's contest were not even born when the pioneers of the UPC teams trained with a (relatively) young Prof. Oak.

What follows is an example of the extreme difficulty of the problems during the good old days. It is from the South Central USA ICPC Regional 2004.

****

Zombies love to eat brains. Yum.

Input

The first line contains a single integer n indicating the number of data sets. The following n lines each represent a data set. Each data set will be formatted according to the following description: A single data set consists of a line X Y, where X is the number of brains the zombie eats and Y is the number of brains the zombie requires to stay alive.

Output

For each data set, there will be exactly one line of output. This line will be "`MMM BRAINS`" if the number of brains the zombie eats is greater than or equal to the number of brains the zombie requires to stay alive. Otherwise, the line will be "`NO BRAINS`".

Sample input

```
3
4 5
3 3
4 3
```

Sample output

```
NO BRAINS
MMM BRAINS
MMM BRAINS
```

****

One observation about the test cases: Assume that both X and Y are between 0 and 10^{10000}.

Problem information

Author : Salvador Roura
Generation : 2022-06-11 12:16:17