Code Jam 2016 - World Finals

Integeregex

Problem

In this problem, a valid regular expression is one of the following. In the following descriptions, E_1, E_2, etc. denote (not necessarily different) valid regular expressions.

- A decimal digit: that is, one of `0 1 2 3 4 5 6 7 8 9`.
- Concatenation: E_1E_2.
- Disjunction: `(`E_1`|`E_2 `|...|`E_N`)`, for at least two expressions. Note that the outer parentheses are required.
- Repetition: `(`E_1`)*`. Note that the outer parentheses are required.

For example, `7`, `23`, `(7)*`, `(45)*`, `(1|2|3)`, `((2)*|3)`, `(1|2|3)`, and `((0|1))*` are valid expressions. `(7)`, `4|5`, `4*`, `(1|)`, and `(0|1)*` are not.

We say that an expression E matches a string of digits D if and only if at least one of the following is true:

- $E = D$.
- $E = E_1E_2$ and there exist D_1 and D_2 such that $D = D_1D_2$ and E_i matches D_i.
- $E =$ `(`E_1`|` E_2`|...|` E_N`)` and at least one of the E_i matches D.
- $E =$ `(`E_1`)*` and there exist D_1, D_2, ..., D_N for some non-negative integer N such that $D = D_1D_2...D_N$ and E_1 matches each of the D_i. In particular, note that `(`E_1`)*` matches the empty string.

For example, the expression `((1|2))*3` matches `3`, `13`, `123`, and `2221123`, among other strings. However, it does *not* match `1234`, `3123`, `12`, or `33`, among other strings.

Given a valid regular expression **R**, for how many integers between **A** and **B**, inclusive, does **R** match the integer's base 10 representation (with no leading zeroes)?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each consists of two lines. The first line has two positive integers **A** and **B**: the inclusive limits of the integer range we are interested in. The second has a string **R** consisting only of characters in the set `0123456789()|*`, which is guaranteed to be a valid regular expression as described in the statement above.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the number of integers in the inclusive range [**A**, **B**] that the the regular expression **R** matches.

Limits

Time limit: 20 seconds per test set.
Memory limit: 1 GB.

$1 \leq \mathbf{T} \leq 100$.
$1 \leq \mathbf{A} \leq \mathbf{B} \leq 10^{18}$.
$1 \leq$ length of $\mathbf{R} \leq 30$.

Small dataset (Test Set 1 - Visible)

R contains no | characters.

Large dataset (Test Set 2 - Hidden)

No additional limits.

Sample

Sample Input

```
8
1 1000
(0)*1(0)*
379009 379009
379009
1 10000
(12)*(34)*
4 5
45
1 100
((0|1))*
1 50
(01|23|45|67|23)
1 10000000000000000000
((0|1|2|3|4|5|6|7|8|9))*
1 1000
1(56|(((7|8))*9)*)
```

Sample Output

```
Case #1: 4
Case #2: 1
Case #3: 5
Case #4: 0
Case #5: 4
Case #6: 2
Case #7: 10000000000000000000
Case #8: 6
```

Note that sample cases 5 through 8 would not appear in the Small dataset.

In sample case 1, the matches in range are 1, 10, 100, and 1000.

In sample case 2, the match in range is 379009.

In sample case 3, the matches in range are 12, 34, 1212, 1234, and 3434.

In sample case 4, there are no matches in range.

In sample case 5, the matches in range are 1, 10, 11, and 100.

In sample case 6, the matches in range are 23 and 45.

In sample case 7, it is possible to form any number in the range.

In sample case 8, the matches in range are 1, 19, 156, 179, 189, and 199.

Code Jam 2016 - World Finals

Family Hotel

Family Hotel

You run a hotel with **N** rooms arranged along one long corridor, numbered from 1 to **N** along that corridor. Your guests are big families, and every family asks for exactly two adjacent rooms when they arrive. Two rooms are adjacent if their numbers differ by exactly 1.

At the start of the day today, your hotel was empty. You have been using the following simple strategy to assign rooms to your guests. As each family arrives, you consider all possible pairs of adjacent rooms that are both free, pick one of those pairs uniformly at random, and assign the two rooms in that pair to the family. New families constantly arrive, one family at a time, but once there are no more pairs of adjacent rooms that are both free, you turn on the NO VACANCY sign and you do not give out any more rooms.

Given a specific room number, what is the probability that it will be occupied at the time that you turn on the NO VACANCY sign?

Input

The first line of the input gives the number of test cases, **T**. **T** lines follow. Each line contains two numbers: the number of rooms **N** and the room number **K** that we are interested in.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the sought probability computed *modulo* 10^9+7, which is defined precisely as follows. Represent the probability that room **K** is occupied as an irreducible fraction `p/q`. The number `y` then must satisfy the modular equation $y \times q \equiv p \pmod{10^9+7}$, and be between 0 and 10^9+6, inclusive. It can be shown that under the constraints of this problem such a number `y` always exists and is uniquely determined.

Limits

Time limit: 20 seconds per test set.
Memory limit: 1 GB.
$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{K} \le \mathbf{N}$.

Small dataset (Test Set 1 - Visible)

$2 \le \mathbf{N} \le 10^4$.

Large dataset (Test Set 2 - Hidden)

$2 \le \mathbf{N} \le 10^7$.

Sample

Sample Input

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

Sample Output

```
Case #1: 500000004
Case #2: 1
Case #3: 666666672
Case #4: 1
```

In sample case #3, there are four rooms and we are looking for probability that the first room is occupied. When the first family arrives, there are 3 possible situations, each with probability 1/3: occupy rooms 1+2, 2+3 or 3+4. In the first situation, the first room is already occupied and will stay occupied. In the second situation, the first room is free and no more families can be accommodated, so it will stay free. Finally, in the third situation, the next arriving family will definitely get rooms 1+2, and thus the first room will be occupied. The probability that the first room is occupied is thus 2/3, and the answer is 666666672, since `(666666672 * 3) mod 1000000007 = 2 mod 1000000007`.

The probability for sample case #1 is 1/2, and for sample cases #2 and #4 it is 1.

Code Jam 2016 - World Finals

Gallery of Pillars

Problem

Your friend Cody-Jamal is working on his new artistic installment called "Gallery of Pillars". The installment is to be exhibited in a square gallery of **N** by **N** meters. The gallery is divided into $\mathbf{N}^2$ squares of 1 by 1 meter, forming an **N** by **N** matrix. The exact center of the southwest corner cell is called the *viewpoint*; a person viewing the artwork is supposed to stand there. Each other cell contains a cylindrical pillar. All pillars have two circular bases of radius **R**: one resting on the floor, in the center of its corresponding cell, and the other touching the gallery's ceiling. The observer will stand in the viewpoint, observe the $\mathbf{N}^2$ - 1 pillars, and marvel.

Cody-Jamal is currently scouting venues trying to see how large he can make the value of **N**. Also, he has not decided which material the pillars will be made of; it could be concrete, or carbon nanotubes, so the radius **R** of the base of each pillar could vary from 1 micrometer to almost half a meter. Notice that a radius of half a meter would make neighboring pillars touch.

You, as a trained mathematician, quickly observe that there could be pillars impossible to see from the viewpoint. Cody-Jamal asks your help in determining, for different combinations of **N** and **R**, the number of visible pillars. Formally, a pillar is visible if and only if there is a straight line segment that runs from the center of the southwest corner cell (the viewpoint) to any point on the pillar's boundary, and does not touch or intersect any other pillar.

Input

The first line of the input gives the number of test cases, **T**. **T** lines follow. Each line describes a different test case with two integers **N** and **R**. **N** is the number of 1 meter square cells along either dimension of the gallery, and **R** is the radius of each pillar, in micrometers. Thus, **R** / 10^6 is the radius of each pillar in meters.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the number of pillars in the installment that are visible from the viewpoint.

Limits

Time limit: 20 seconds per test set.
Memory limit: 1 GB.
$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{R} < 10^6 / 2$.

Small dataset (Test Set 1 - Visible)

$2 \le \mathbf{N} \le 300$.

Large dataset (Test Set 2 - Hidden)

$2 \le \mathbf{N} \le 10^9$.

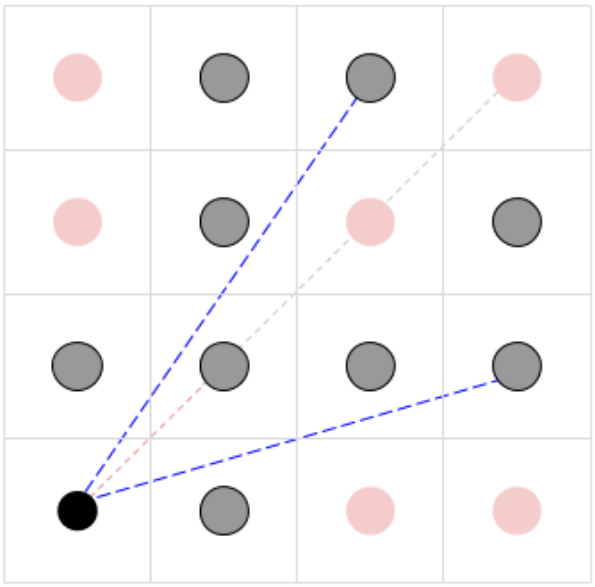
Sample

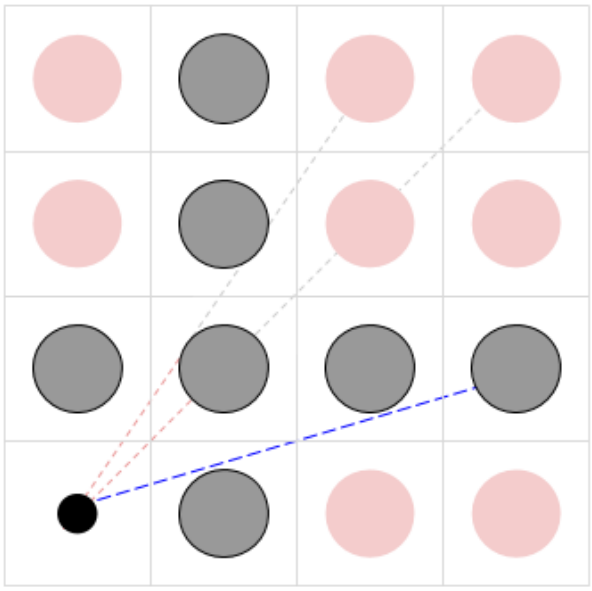
Sample Input	Sample Output
4 4 100000 4 300000	Case #1: 9 Case #2: 7

```
3 300000
100 499999
```

```
Case #3: 5
Case #4: 3
```

The pictures below illustrate the first two samples (not to scale). In the center of the black circle is the observer. The other circles are pillars, with the visible ones in gray and the not visible ones in red. The blue dotted lines represent some of the unblocked lines of sight; the red dotted lines represent blocked lines of sight (that turn gray at the point at which they are first blocked).





Code Jam 2016 - World Finals

Map Reduce

Problem

Ben the brilliant video game designer is trying to design maps for his upcoming augmented-reality mobile game. Recently, he has created a map which is represented as a matrix of **R** rows and **C** columns. The map consists of a bunch of `.` characters representing empty squares, a bunch of `#` characters representing impassable walls, a single start position represented by `S` and a single finish position represented by `F`. For example, the map could look like:

```
############
#S..#..##...#
###.##..#.#F#
#...##.##.###
#.#.........#
############
```

In Ben's game, a *path* is a sequence of steps (up, down, left or right) to go from one cell to another while not going through any impassable walls.

Ben considers a *good* map to have the following properties:

- There is a path between any two empty squares (including the start and finish positions).
- To preserve structural integrity, impassable walls must meet at edges and not just corners. For every 2×2 region in the map, if the region contains exactly two walls, then those walls are either in the same row or the same column. In other words, there is no 2×2 region where the walls are in one of these two configurations:

  ```
  #.    .#
  .#    #.
  ```

- The boundary consists of only impassable walls. A cell is considered part of the boundary if it is in the uppermost/lowermost rows or if it is in the leftmost/rightmost columns.

The distance of the shortest path is the minimum number of steps required to reach the finish position from the start position. For instance, the shortest path in the above example takes 17 steps.

Being such a clever mapmaker, Ben realized that he has created a map that is too hard for his friends to solve. He would like to reduce its difficulty by removing some of the impassable walls. In particular, he wants to know whether it is possible to remove zero or more impassable walls from his map such that the shortest path from start to finish takes *exactly* **D** steps, and that the resulting map is still *good*. Note that it is not enough to simply find *a* path with **D** steps; **D** must be the number of steps in the *shortest* path.

For example, if **D** = 15, we could remove the impassable wall directly below the finish position to get a good solution.

```
############
#S..#..##...#
###.##..#.#F#
#...##.##.#.#
```

```
#.#.........#
############
```

There is no solution if **D** = 5.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing three space-separated integers **R**, **C** and **D**: the number of rows and columns in the map, and the desired number of steps in the shortest path from start to finish after possibly removing impassable walls. **R** lines follow, each consisting of **C** characters (either `.`, `#`, `S` or `F`) representing Ben's map.

It is guaranteed that the map is good, as described in the problem statement.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the word `POSSIBLE` or `IMPOSSIBLE`, depending on whether the shortest path can be made equal to **D** by removing some number of walls such that the map is still good. If it is possible, output **R** more lines containing **C** characters each, representing the new map. In your output, replace the `#` characters for removed walls (if any) with `.` characters.

If multiple solutions are possible, you may output any of them.

Limits

Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$.
Each test case contains exactly one `S` and exactly one `F`.
The input file is at most 3MB in size.

Small dataset (Test Set 1 - Visible)

Time limit: 60 seconds.
$3 \leq \mathbf{R} \leq 40$.
$3 \leq \mathbf{C} \leq 40$.
$1 \leq \mathbf{D} \leq 1600$.

Large dataset (Test Set 2 - Hidden)

Time limit: 300 seconds.
$3 \leq \mathbf{R} \leq 1000$.
$3 \leq \mathbf{C} \leq 1000$.
$1 \leq \mathbf{D} \leq 10^6$.

NOTE: The Large output breaks the usual cap on Code Jam output size, but you can upload it as normal.

Sample

Sample Input	Sample Output

```
3
6 13 15
#############
#S..#..##...#
###.##..#.#F#
#...##.##.###
#.#.........#
#############
5 8 3
########
#S.....#
####...#
#F.....#
########
4 10 11
##########
#S#...#.F#
#...#...##
##########
```

```
Case #1: POSSIBLE
#############
#S..#..##...#
###.##..#.#F#
#...##.##.#.#
#.#.........#
#############
Case #2: IMPOSSIBLE
Case #3: POSSIBLE
##########
#S#...#.F#
#...#...##
##########
```

The sample output displays one set of answers to the sample cases. Other answers may be possible.

Sample case #1 is the example in the problem statement.

In sample case #2, it is possible to remove walls to make the distance of the shortest path either 2 or 4, for example. However, there is no way to make the distance of the shortest path exactly 3.

In sample case #3, the shortest path already takes 11 steps to begin with, so there is no need to reduce the difficulty of the map.

Code Jam 2016 - World Finals

Radioactive Islands

Problem

You are steering a boat from the coordinates (-10, **A**) to the coordinates (10, **B**). The coordinates are measured in kilometers, and your boat travels at a constant speed of 1 kilometer per hour. You have full control over the path the boat takes. We model the boat as a single point.

There are **N** islands in the area; we model them as single points. The i-th island is at the coordinates (0, $\mathbf{C_i}$).

The area is radioactive, and you constantly receive 1 microsievert per hour of radiation from the general environment, no matter where you are. Moreover, the islands themselves are radioactive, and you constantly receive additional radiation at a rate of $(D_i)^{-2}$ microsieverts per hour from the i-th island, where D_i is your current distance (in kilometers) from the i-th island. (Formally: let $D_i(t)$ be your distance from the i-th island as a function of time t, and X be the total time your journey takes. Then the total radiation received from the i-th island is the definite integral from 0 to X of $D_i(t)^{-2}$.) You can get as close to an island as you would like, as long as you do not match its exact coordinates.

Find the minimum total radiation dose that you can receive if you plot your course optimally.

Input

The first line of the input gives the number of test cases, **T**; **T** test cases follow. Each test cases consists of two lines. The first line of a test case consists of three values: an integer **N**, and two floating-point numbers **A** and **B**, as described in the statement above. The second line of a test case consists of **N** floating-point numbers $\mathbf{C_i}$; the i-th of these numbers gives the y coordinate of the i-th island.

All floating-point numbers are specified to exactly two decimal places.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the minimum radiation dose (in microsieverts) received while completing the journey.

`y` will be considered correct if it is within an absolute or relative error of 10^{-3} of the correct answer. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits

Memory limit: 1 GB.
-10.00 ≤ **A** ≤ 10.00.
-10.00 ≤ **B** ≤ 10.00.
-10.00 ≤ $\mathbf{C_i}$ ≤ 10.00, for all i.
$\mathbf{C_i} \neq \mathbf{C_j}$, for all i ≠ j.

Small dataset (Test Set 1 - Visible)

Time limit: 120 seconds.
T ≤ 20;
N = 1.

Large dataset (Test Set 2 - Hidden)

Time limit: 240 seconds.
T ≤ 50;
1 ≤ **N** ≤ 2.

Sample

Sample Input	Sample Output
2 1 1.00 -2.00 0.00	Case #1: 21.806 Case #2: 21.706

```
2 0.00 0.00
3.00 -3.00
```

Here is a diagram of the optimal path for sample case #1. We have enlarged the island to make it more visible, but remember to treat it as a single point.