Problem 640: Solve the Equation

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Solve a given equation and return the value of

'x'

in the form of a string

"x=#value"

. The equation contains only

'+'

,

'-'

operation, the variable

'x'

and its coefficient. You should return

"No solution"

if there is no solution for the equation, or

"Infinite solutions"

if there are infinite solutions for the equation.

If there is exactly one solution for the equation, we ensure that the value of

'x'

is an integer.

Example 1:

Input:

equation = "x+5-3+x=6+x-2"

Output:

"x=2"

Example 2:

Input:

equation = "x=x"

Output:

"Infinite solutions"

Example 3:

Input:

equation = "2x=x"

Output:

"x=0"

Constraints:

3 <= equation.length <= 1000

equation

has exactly one

'='

.

equation

consists of integers with an absolute value in the range

[0, 100]

without any leading zeros, and the variable

'x'

.

The input is generated that if there is a single solution, it will be an integer.

Code Snippets

C++:

```
class Solution {
public:
string solveEquation(string equation) {

}
};
```

Java:

```
class Solution {
public String solveEquation(String equation) {

}
}
```

Python3:

```
class Solution:
def solveEquation(self, equation: str) -> str:
```

Python:

```
class Solution(object):
def solveEquation(self, equation):
"""
:type equation: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} equation
* @return {string}
*/
var solveEquation = function(equation) {

};
```

TypeScript:

```
function solveEquation(equation: string): string {

};
```

C#:

```
public class Solution {
public string SolveEquation(string equation) {

}
}
```

C:

```
char* solveEquation(char* equation) {

}
```

Go:

```
func solveEquation(equation string) string {

}
```

Kotlin:

```
class Solution {
fun solveEquation(equation: String): String {

}
}
```

Swift:

```
class Solution {
func solveEquation(_ equation: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn solve_equation(equation: String) -> String {

}
}
```

Ruby:

```
# @param {String} equation
# @return {String}
def solve_equation(equation)

end
```

PHP:

```
class Solution {

/**
* @param String $equation
* @return String
*/
function solveEquation($equation) {

}
}
```

Dart:

```
class Solution {
String solveEquation(String equation) {

}
}
```

Scala:

```
object Solution {
def solveEquation(equation: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec solve_equation(equation :: String.t) :: String.t
def solve_equation(equation) do

end
end
```

Erlang:

```
-spec solve_equation(Equation :: unicode:unicode_binary()) ->
unicode:unicode_binary().
solve_equation(Equation) ->
```

```
.
```

Racket:

```
(define/contract (solve-equation equation)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Solve the Equation
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
string solveEquation(string equation) {

}
};
```

Java Solution:

```
/**
* Problem: Solve the Equation
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
class Solution {
public String solveEquation(String equation) {

}
}
```

Python3 Solution:

```
"""
Problem: Solve the Equation
Difficulty: Medium
Tags: string, math

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def solveEquation(self, equation: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def solveEquation(self, equation):
"""
:type equation: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Solve the Equation
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} equation
* @return {string}
*/
var solveEquation = function(equation) {

};
```

TypeScript Solution:

```
/**
* Problem: Solve the Equation
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function solveEquation(equation: string): string {

};
```

C# Solution:

```
/*
* Problem: Solve the Equation
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
```

```
public string SolveEquation(string equation) {

}
}
```

C Solution:

```
/*
* Problem: Solve the Equation
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char* solveEquation(char* equation) {

}
```

Go Solution:

```
// Problem: Solve the Equation
// Difficulty: Medium
// Tags: string, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func solveEquation(equation string) string {

}
```

Kotlin Solution:

```
class Solution {
fun solveEquation(equation: String): String {

}
```

```
}
```

Swift Solution:

```
class Solution {
func solveEquation(_ equation: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Solve the Equation
// Difficulty: Medium
// Tags: string, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn solve_equation(equation: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} equation
# @return {String}
def solve_equation(equation)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $equation
* @return String
```

```
*/
function solveEquation($equation) {

}
}
```

Dart Solution:

```
class Solution {
String solveEquation(String equation) {

}
}
```

Scala Solution:

```
object Solution {
def solveEquation(equation: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec solve_equation(equation :: String.t) :: String.t
def solve_equation(equation) do

end
end
```

Erlang Solution:

```
-spec solve_equation(Equation :: unicode:unicode_binary()) ->
unicode:unicode_binary().
solve_equation(Equation) ->
.
```

Racket Solution:

```
(define/contract (solve-equation equation)
(-> string? string?)
)
```