Problem 3484: Design Spreadsheet

Problem Information

Difficulty: Medium
Acceptance Rate: 74.63%
Paid Only: No
Tags: Array, Hash Table, String, Design, Matrix

Problem Description

A spreadsheet is a grid with 26 columns (labeled from `'A'` to `'Z'`) and a given number of `rows`. Each cell in the spreadsheet can hold an integer value between 0 and 105.

Implement the `Spreadsheet` class:

* `Spreadsheet(int rows)` Initializes a spreadsheet with 26 columns (labeled `'A'` to `'Z'`) and the specified number of rows. All cells are initially set to 0. * `void setCell(String cell, int value)` Sets the value of the specified `cell`. The cell reference is provided in the format `"AX"` (e.g., `"A1"`, `"B10"`), where the letter represents the column (from `'A'` to `'Z'`) and the number represents a **1-indexed** row. * `void resetCell(String cell)` Resets the specified cell to 0. * `int getValue(String formula)` Evaluates a formula of the form `"=X+Y"`, where `X` and `Y` are **either** cell references or non-negative integers, and returns the computed sum.

Note: If `getValue` references a cell that has not been explicitly set using `setCell`, its value is considered 0.

Example 1:

Input: ["Spreadsheet", "getValue", "setCell", "getValue", "setCell", "getValue", "resetCell", "getValue"] [[3], ["=5+7"], ["A1", 10], ["=A1+6"], ["B2", 15], ["=A1+B2"], ["A1"], ["=A1+B2"]]

Output: [null, 12, null, 16, null, 25, null, 15]

Explanation

Spreadsheet spreadsheet = new Spreadsheet(3); // Initializes a spreadsheet with 3 rows and 26 columns spreadsheet.getValue("=5+7"); // returns 12 (5+7) spreadsheet.setCell("A1", 10); // sets A1 to 10 spreadsheet.getValue("=A1+6"); // returns 16 (10+6) spreadsheet.setCell("B2", 15); // sets B2 to 15 spreadsheet.getValue("=A1+B2"); // returns 25 (10+15) spreadsheet.resetCell("A1"); // resets A1 to 0 spreadsheet.getValue("=A1+B2"); // returns 15 (0+15)

Constraints:

* `1 <= rows <= 103` * `0 <= value <= 105` * The formula is always in the format `"=X+Y"`, where `X` and `Y` are either valid cell references or **non-negative** integers with values less than or equal to `105`. * Each cell reference consists of a capital letter from `'A'` to `'Z'` followed by a row number between `1` and `rows`. * At most `104` calls will be made in **total** to `setCell`, `resetCell`, and `getValue`.

Code Snippets

C++:

```
class Spreadsheet {
public:
Spreadsheet(int rows) {

}

void setCell(string cell, int value) {

}

void resetCell(string cell) {

}

int getValue(string formula) {

}
};

/**
* Your Spreadsheet object will be instantiated and called as such:
* Spreadsheet* obj = new Spreadsheet(rows);
```

```
* obj->setCell(cell,value);
* obj->resetCell(cell);
* int param_3 = obj->getValue(formula);
*/
```

Java:

```
class Spreadsheet {

public Spreadsheet(int rows) {

}

public void setCell(String cell, int value) {

}

public void resetCell(String cell) {

}

public int getValue(String formula) {

}
}

/**
* Your Spreadsheet object will be instantiated and called as such:
* Spreadsheet obj = new Spreadsheet(rows);
* obj.setCell(cell,value);
* obj.resetCell(cell);
* int param_3 = obj.getValue(formula);
*/
```

Python3:

```
class Spreadsheet:

def __init__(self, rows: int):


def setCell(self, cell: str, value: int) -> None:
```

```
def resetCell(self, cell: str) -> None:


def getValue(self, formula: str) -> int:



# Your Spreadsheet object will be instantiated and called as such:
# obj = Spreadsheet(rows)
# obj.setCell(cell,value)
# obj.resetCell(cell)
# param_3 = obj.getValue(formula)
```