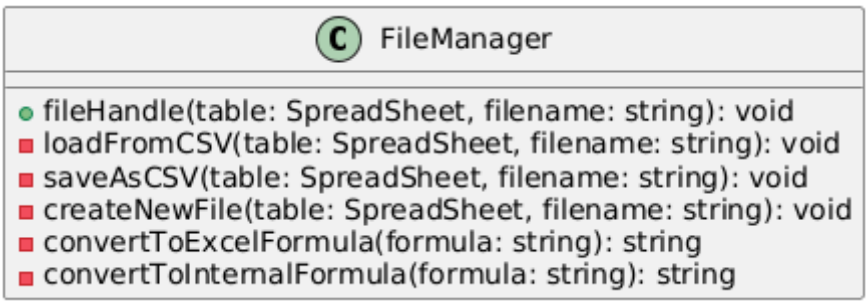
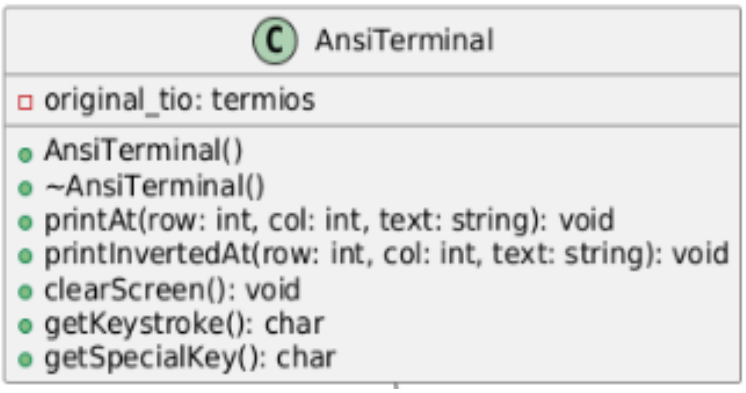
ANSI Terminal-Based Spreadsheet Program

1-UML Class Diagrams:

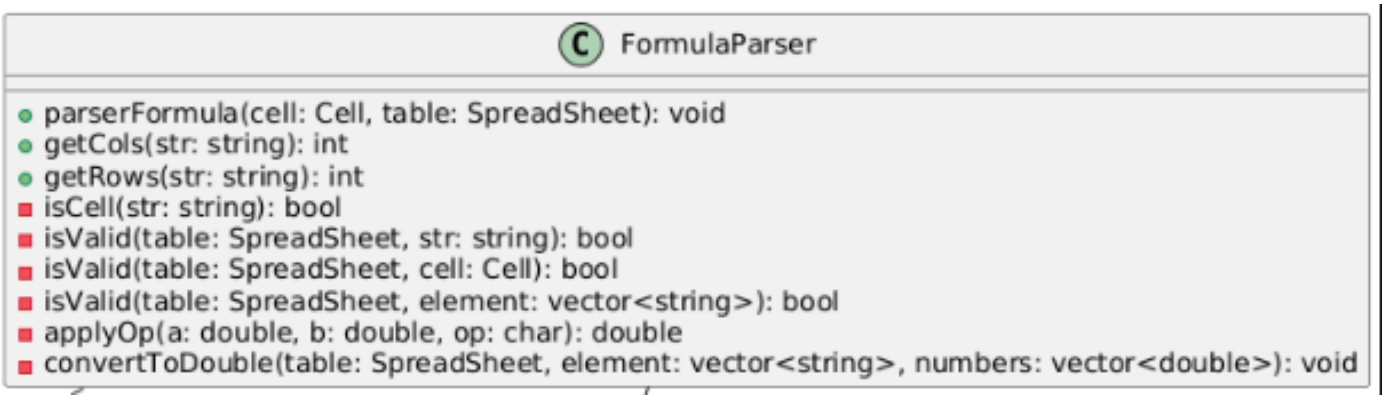
1.1-FileManager class :



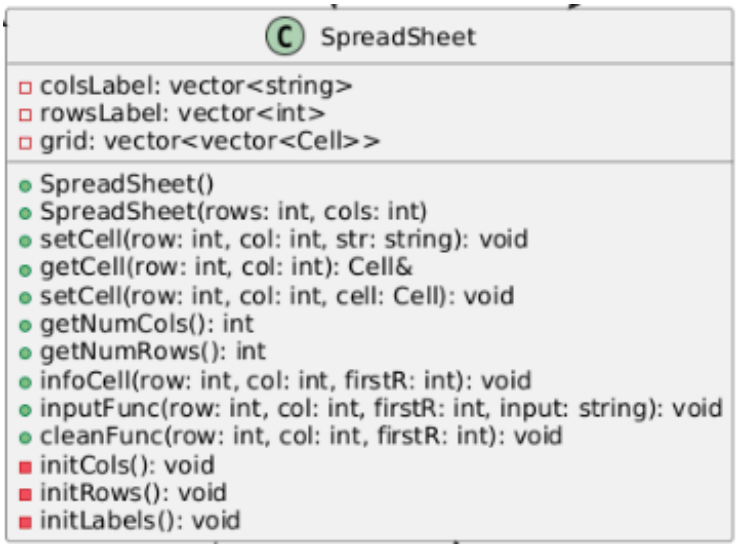
1.2- AnsiTerminal class :



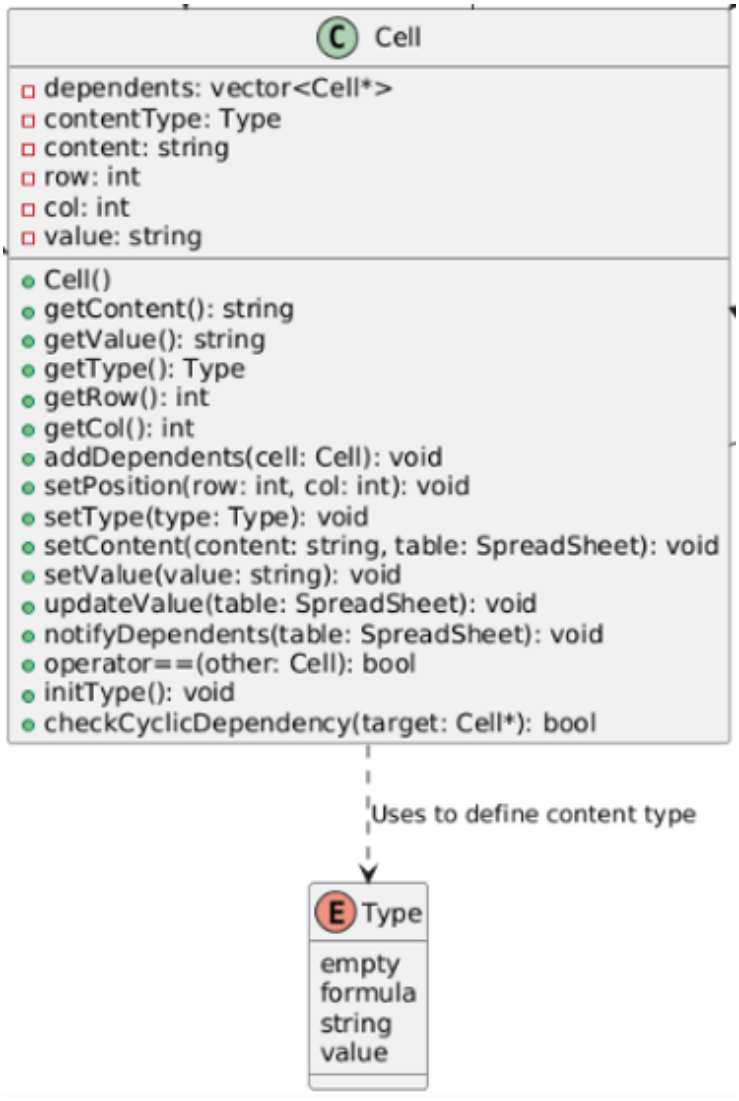
1.3-FormulaParser class:



1.4-SpreadSheet class :



1.5-Cell class :



The relationship of all classes to each other :

2-Description of implemented features and missing features:

2.1 Automatic Cell Calculation :

Description:

-> Cells automatically calculate and update their values when their referenced cells change.

-> This is achieved through the notifyDependents function in the Cell class, which propagates changes to dependent cells.

Implementation:

-> When a cell's content is updated, its dependents are notified, and their formulas are recalculated.

-> This ensures that changes ripple through the spreadsheet without manual intervention.

Example:

-> Consider A1 = 10 and B1 = A1 * 2.

-> When A1 is updated to 20, the notifyDependents mechanism automatically updates B1 to 40.

2.2 Cell Referencing :

Description:

-> A cell can reference other cells using cell identifiers (e.g., A1, B2, etc.).

-> This allows formulas to use values from other cells, either directly or as part of calculations.

Implementation:

-> The FormulaParser handles cell referencing by extracting row and column numbers (getRows, getCols) from the reference string.

-> If the referenced cells change, their dependents (like the formula cell) are updated.

Example:

-> In C1 = A1 + B1, changing A1 or B1 recalculates C1 dynamically.

-> If A1 = 5 and B1 = 15, C1 becomes 20. Updating A1 to 10 makes C1 automatically update to 25.

2.3 Range-Based Functions :

Description:

-> Functions like SUM, AVER, MAX, MIN.CPY, and STDDEV operate over a range of cells.

-> The syntax for ranges is @FUNCTION(A1..A5) except CPY.

-> The syntax for CPY function is <A1-CPY(B1..B7). In this formula, A1 indicates the cell to be copied, and the cells in parentheses indicate where it will be copied.

Implementation:

-> The parserFormula function parses ranges, converts cell values to doubles, and applies the function.

-> Results are calculated dynamically and updated in the target cell.

Example:

-> @SUM(A1..A3) adds values in cells A1 to A3.

-> @SUM(A1..C1) adds values in cells A1 to C1.

-> <D1-CPY(A1..A5) The contents of cell D1 are copied from A1 to A5.

2.4 Arithmetic Operations:

Description:

-> Formulas begin with ' =' and supports basic arithmetic (+, -, *, /) operations.

Implementation:

-> The FormulaParser::parserFormula function interprets and evaluates formulas.

-> It separates components (operators, numbers, or cell references) and processes them based on their type.

Example:

-> =A1 + B2 * 3:

If A1 = 5 and B2 = 2, the result is 5 + (2 * 3) = 11.

-> =-A1*-B1+C3:

If A1=2, B1=4 and C3=5, the result is (-2*-4)+5=13;

2.5 User Commands

Description:

-> Commands allow users to navigate, edit, or perform operations directly via the terminal.

Commands:

-> Arrow Keys (U/D/L/R): Navigate the grid.

-> '>': Jump to a cell. Example: >B3 moves the cursor to cell B3.

-> '&': Save the spreadsheet. Example: &SAVE filename saves the spreadSheet to filename cvs, &LOAD filename uploads data from csv file to spreadsheet and &NEW filename creates new file.

-> ~RESET: Clears all cells.

-> @SUM, @AVER, etc.: Perform range calculations.

-> '=': Perform arithmetic operations.

-> 'q' : Terminates the program.

2.6 Missing Features:

-> The program only supports columns from AA to ZZ, for example AAA is not supported.

-> Maybe there are error checks not taken into account.

-> Since there is no horizontal scrolling, care must be taken to enter as many columns as will fit on the screen, otherwise the program may not work properly.

-> While entering a string, the keys **U, D, R,** and **L** are interpreted as directional keys, so they cannot be used as input and are only for navigation. Similarly, the **J** key is interpreted as the Enter key, so it also cannot be used as input.

3- Declaration of AI assistance

```
// Loads spreadsheet content from a CSV file.
void FileManager::loadFromCSV(SpreadSheet& table, const string& filename) {
    ifstream file(filename);
    if (!file.is_open()) {
        // If the file cannot be opened, return early.
        return;
    }

    string line;
    int row = 0;
    // Read the file line by line.
    while (getline(file, line)) {
        istringstream ss(line);
        string cellContent;
        int col = 0;

        // Split the line by commas to get cell contents.
        while (getline(ss, cellContent, ',')) {
            table.getCell(row, col).setContent(cellContent, table);
            ++col;
        }

        // If there are fewer columns than the spreadsheet's total columns,
        // set the remaining cells in the row to empty.
        if (col < table.getNumCols()) {
            table.getCell(row, col).setContent("", table);
        }
        ++row;
    }
```

-> Since I did not know how to use istringstrream in this part, I got help from artificial intelligence there.

```cpp
bool Cell::checkCyclicDependency(Cell* target) {
    // Check if the current cell is the same as the target cell.
    // If so, it means a cyclic dependency is detected (cell depends on itself).
    if (this == target) {
        return true; // A cyclic dependency exists.
    }

    // Iterate through all dependent cells of the current cell.
    for (Cell* dependent : dependents) {
        // Recursively check each dependent cell for a cyclic dependency.
        if (dependent->checkCyclicDependency(target)) {
            return true; // A cyclic dependency has been found.
        }
    }

    return false; // No cyclic dependency found.
}



// Function to notify all dependent cells that the value of the current cell has changed
void Cell::notifyDependents(SpreadSheet& spreadsheet) {
    if (dependents.size() != 0) {
        for (Cell* dep : dependents) {
            if(dep->getType()==Type::formula){
                dep->updateValue(spreadsheet); // Update the value of each dependent cell
            }
            else{
                dep->dependents.clear();
            }
        }
    }
}
```

-> In this part, since I did not know how to prevent the cyclec dependency and in addition, I got the idea of the notifydependesntes function, which I used for automatic calculation, from artificial intelligence.

```cpp
   // Process multiplication and division first
   for (int i = 0; i < op.size(); i++) {
       if (op[i] == '*' || op[i] == '/') {
           numbers[i] = applyOp(numbers[i], numbers[i + 1], op[i]);  // Apply operation
           numbers.erase(numbers.begin() + i + 1);  // Remove the processed number
           op.erase(op.begin() + i);  // Remove the operator
           i--;  // Adjust the index after removal
       }
   }

   // Process remaining addition and subtraction operations
   while (!op.empty()) {
       double result = applyOp(numbers[0], numbers[1], op[0]);  // Apply the operation
       numbers[0] = result;  // Store the result back
       numbers.erase(numbers.begin() + 1);  // Remove the second operand
       op.erase(op.begin());  // Remove the operator
   }
```

-> In this section, I deleted the relevant element of the vectors that hold the numbers and operators that we will use in arithmetic calculations after the calculated value, with the help of artificial intelligence.

```cpp
string FileManager::convertToExcelFormula(const string& formula) {
    vector<string> originalFunctions = {"@SUM", "@MAX", "@MIN", "@AVER", "@STDDEV"};
    vector<string> excelFunctions = {"=SUM", "=MAX", "=MIN", "=AVERAGE", "=STDEV"};

    string result = formula;

    // Replace the function name.
    for (int i = 0; i < originalFunctions.size(); ++i) {
        int pos = result.find(originalFunctions[i]);
        if (pos != string::npos) {
            result.replace(pos, originalFunctions[i].length(), excelFunctions[i]);
            break;
        }
    }

    // Replace ".." with ":" for range compatibility in Excel.
    int rangePos = result.find("..");
    if (rangePos != string::npos) {
        result.replace(rangePos, 2, ":");
    }

    return result;
}
```

-> In this section, I will get help from artificial intelligence while making the necessary conversions for Spreashet to be compatible with Excel.

-> And in general, I got help from artificial intelligence in writing the comment lines.

4-User Manual :

4.1 Information you need to know:

-> If you try to enter a string into the cells, the string is written directly into the cell, but if you try to enter a formula or double number, the things you type first appear in the input function section and after you press the Enter key, they appear in the cell.

-> "If you try to perform arithmetic operations with an empty cell or a cell containing a string, the string or empty cells will be treated as 0.0 in double, and operations will be performed accordingly. Additionally, if you attempt to divide by 0, you will encounter an error; in this case, the division will be performed with 1 instead of 0.

For example, if cell A1 contains “CSE”, B1 contains 5, and C1 contains 0:

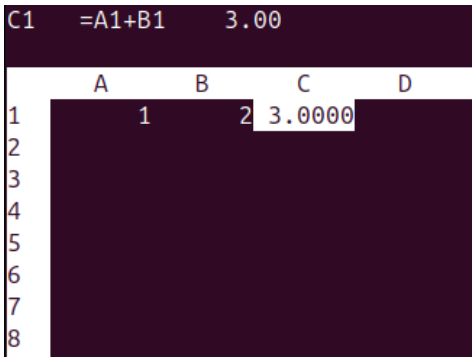
-> A1 + B1 will result in 5, as the string (“CSE:) will be treated as 0 and the operation will be performed with 5.

-> B1 / C1 will also result in 5, since dividing by 0 causes an error, and the division is done with 1 instead."

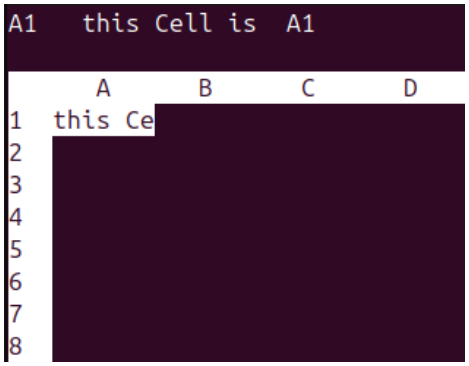
-> Each cell is set to display a maximum of 7 characters. If you enter a string or number longer than this, the full information of the cell will appear in the top row when

you hover over the cell. In this row, the cell's position is displayed first, followed by the formula of the cell (if the cell type is a formula), and finally the cell's content. As shown in the images below.

If cell contains fomrula:



If cell does nor contain formula:

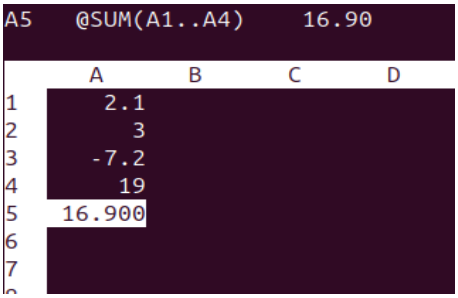


4.2 Use of formulas;

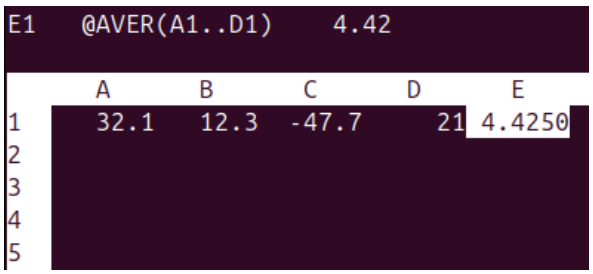
4.2.1 Usage of '@':

-> There are 5 formulas that start with '@': @AVER, @SUM, @MIN, @MAX, and @STDDEV. These formulas perform the necessary operations on cells located next to or below each other within a specified range. For example, @SUM(A1..A9) calculates the sum of the cells from A1 to A9, or @AVER(A1..D1) calculates the average of the cells from A1 to D1 and assigns the result to the value of the cell. Examples are shown in the images below.

A-) A5=@SUM(A1..A4) = A1+A2+A3+A4



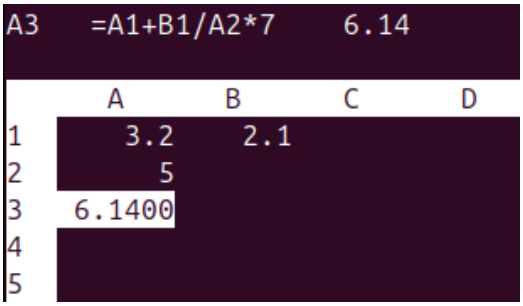
B-) E1=@AVER(A1..D1)=(A1+B1+C1+D1)/4

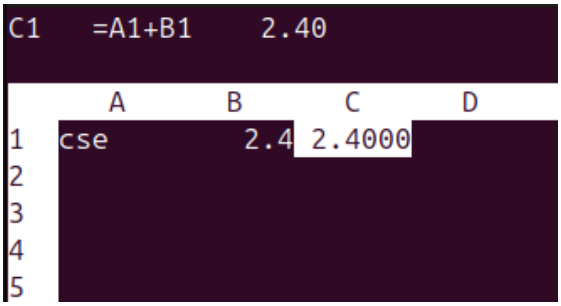


4.2.2 Usage of '=':

-> The cell calculates and assigns the result of the input entered in the cell. Strings, numbers and cell references can be used as input. String cells will be treated as 0.0 in double. For example, if A1 = 2 and B1 = -1, and the formula =A1+B1*3 is entered in a cell, it will calculate the operation (=2+-1*3) and assign the result of -1 to the cell. Examples are shown in the images below.

the string cells will be treated as 0.0 in double

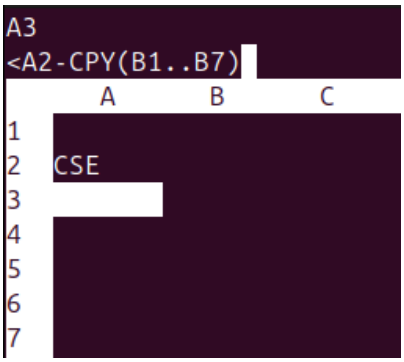




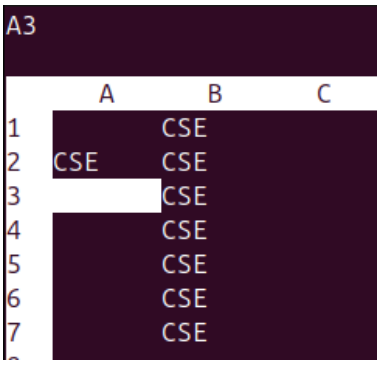
4.2.3 Usage of '<':

-> This sign is only valid for the CPY formula, and as input, it copies the content of the specified first cell and pastes it into all the cells in the given range. The usage is as follows: <C1-CPY(A1..A3) will assign the content of C1 to cells A1, A2, and A3. . Examples are shown in the images below.

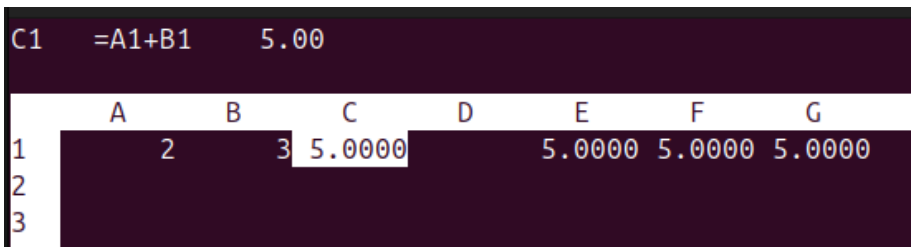
Before pressing enter :



After pressing enter:



<C1-CPY(E1..G1)

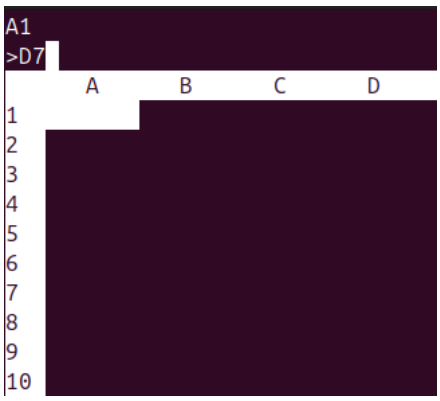


4.3 User commands:

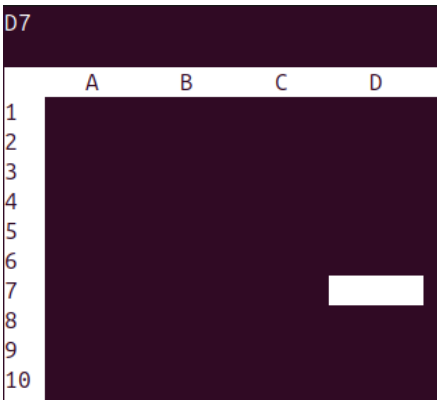
4.3.1 Usage of '>' :

-> It moves the cursor to the specified cell. For example, the >A5 command moves the cursor from its current position to cell A5.

Before pressing enter :



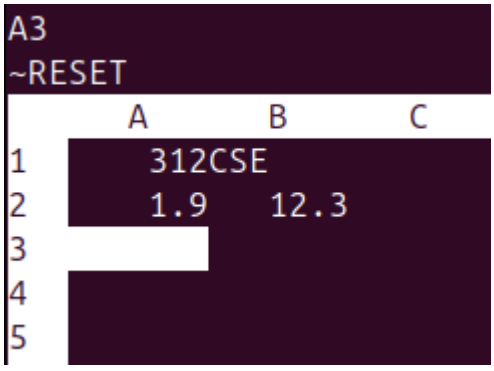
After pressing enter:



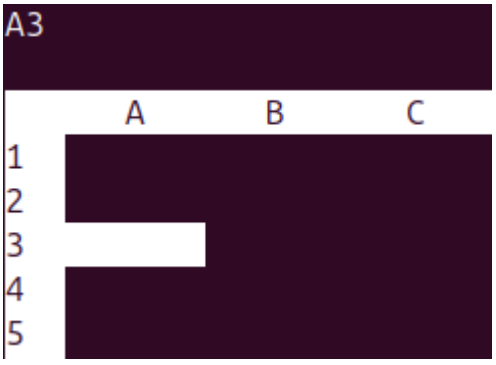
4.3.2 usage of '~'

-> If you enter ~RESET in the input section and press Enter, the entire spreadsheet will be reset."

Before pressing enter :



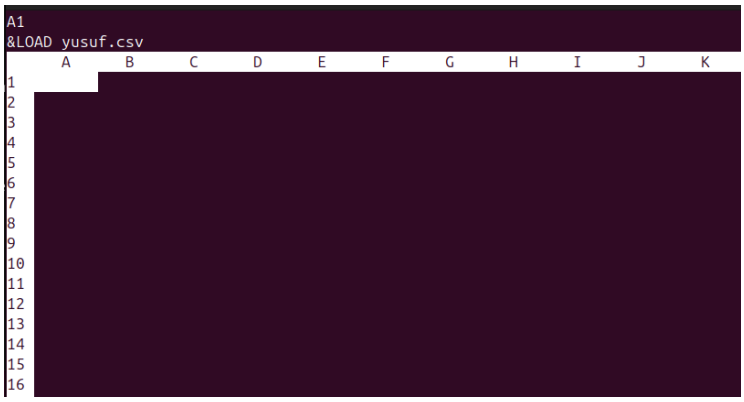
After pressing enter:



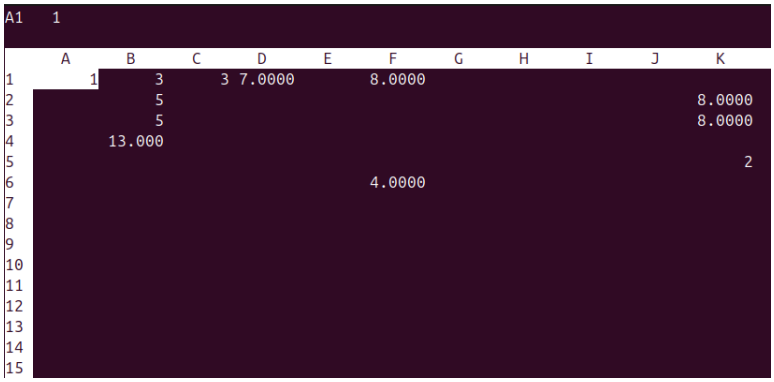
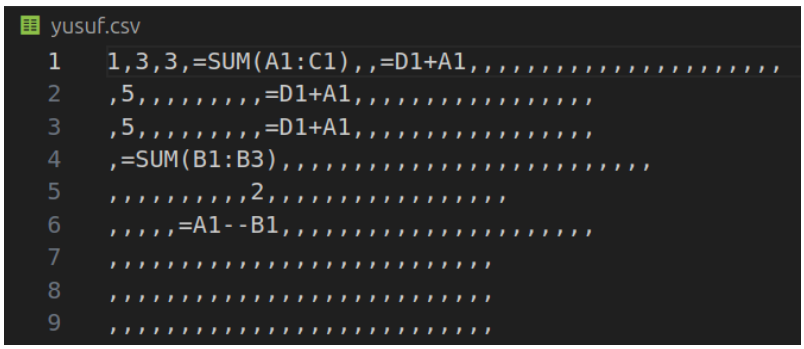
4.3.2 usage of '&':

-> The &SAVE "filename" command saves the spreadsheet to a file named filename. This file should have a .csv extension. The &LOAD "filename" command loads the contents of the filename.csv file into the spreadsheet. The &NEW "filename" command creates a new .csv file with the name filename and & sign makes the necessary conversions for Excel. Examples are shown in the images below.

Before pressing enter :



After pressing enter:



yusuf.csv file;