**Spreadsheet Task Report**

**Brief Description of the Code Structure**

The provided Java application is designed to parse and evaluate a CSV file containing cells with expressions in postfix notation. Here’s a brief overview of its structure:

**Main Method:** The entry point of the application. It validates input arguments, reads the input CSV file, evaluates its content, prints results to STDOUT, and writes the evaluated data to a fixed output file named `output.csv`.

**readCSV Method:** Reads the CSV file specified by the user and stores its contents in a 2D list structure. Each cell in the CSV is read as a string and split by commas.

**evaluateCSV Method:** Iterates through the rows and columns of the CSV data, evaluates each cell’s postfix expression using the `evaluateCell` method, and handles errors gracefully.

**evaluateCell Method:** Handles the evaluation of individual cells by processing postfix expressions, managing cell references, and performing arithmetic operations. It uses a stack-based approach to compute results and checks for circular references and invalid expressions.

**isNumeric Method:** Checks if a string represents a numeric value.

**isCellReference Method:** Determines if a string is a cell reference (e.g., `A1`, `B2`).

**getCellValue Method:** Retrieves the value of a cell based on its reference.

**formatValue Method:** Formats numeric values for output, ensuring integers are printed without decimal places and floating-point numbers are formatted to one decimal place.

**printCSV Method:** Prints the evaluated CSV data to STDOUT.

**writeCSV Method:** Writes the evaluated CSV data to a fixed output file named `output.csv`.

**Limitations and Considerations**

**Cell Reference Handling:** The implementation assumes that cell references are valid and within bounds. It does not include explicit checks for out-of-bounds references or missing rows/columns, which might lead to `IndexOutOfBoundsException` if the input CSV references non-existent cells.

**Error Handling:** Although errors are captured and `#ERR` is added to the CSV output, specific error messages are not provided in the output file. More detailed error reporting could improve debugging and user feedback.

**Postfix Expression Validity:** The evaluation method assumes well-formed postfix expressions. It does not handle cases where expressions are incomplete or contain errors beyond simple invalid operators or insufficient operands.

**Performance:** The approach to evaluating cells could become inefficient with large CSV files or deeply nested cell references. Optimization might be needed for handling larger datasets or more complex expressions.

**Assumptions:** The code assumes that cell references are in uppercase (e.g., `A1`), and it only supports single-letter column references. Support for multiple-letter column references (e.g., `AA1`) is not included.

This implementation meets the basic requirements of parsing and evaluating postfix expressions in a CSV file but could be enhanced with additional error handling, performance optimizations, and support for extended features.