DATA SYSTEM- M'23

Project Phase 1

Due Date : `Saturday, September 2nd, 2023 EOD`

Instructions

- This is a Team Project. You will be working on this project in a team as announced before.
- All commits are to be done in the `master` branch of the repository which is added to your team on github classroom. You may create separate branches, but you are supposed to merge it into the master branch by the due date.
- You must output the following statements (note: case sensitive) onto the terminal, for commands dealing with blocks:
  - `Number of blocks read: <number>`
  - `Number of blocks written: <number>`
  - `Number of blocks accessed: <number>`
    - PS: `blocks accessed = blocks read + blocks written`
- Late Day Form to be filled on the day of submission, if any commit is made past the due date in the `master` branch. You are supposed to fill the correct date of the last commit as we will be using that to download your team's repository from github.
- No personal queries will be entertained. All queries regarding Project Phase 1 is to be posted on the doubts document.
- Tasks 1 and 2 are already submitted as part of Lab Session. Make sure you have create the query files in the data folder and named it correctly: `data/{a-h}.ra`
- Not following any of the above instructions will attract penalty.
- Following tasks are part of Project Phase 1 and was submitted during Lab Session:
  - Queries
  - `SOURCE` Command

TASK: Handle Matrices

Extend SimpleRA to handle Matrices. The application currently supports only tables. Unlike tables, matrices do not have a column name. As it is an integer only application, the values of the matrix will be integers. Square `n x n, (n <= 10^4)` matrices will be provided as `.csv` files in the `data` folder. For instance, Given matrix `A` :

$$\begin{bmatrix} 9 & 13 & 5 & 2 \\ 1 & 11 & 7 & 6 \\ 3 & 7 & 4 & 1 \\ 6 & 0 & 7 & 10 \end{bmatrix}$$

To load A into the system, a file called A.csv should be present in the data folder with the following contents

```
9,13,5,2
1,11,7,6
3,7,4,1
6,0,7,10
```

Note

- Unlike tables, the first line of A.csv represents the first row of the matrix, NOT the column names.
- Each line represents a row of the matrix.
- Values in the rows are comma(,) separated.

Commands

Assume a file named A.csv present in the data folder.

You will implement the following commands to deal with Matrices:

```
LOAD MATRIX
```

Syntax: LOAD MATRIX < matrix_name >

Example: LOAD MATRIX A

Function: Reads a file named A.csv from the data folder and loads into memory as a matrix named A.

```
PRINT MATRIX
```

Syntax: PRINT MATRIX < matrix_name >

Example: PRINT MATRIX A

Function: Similar to PRINT command in case of tables, the PRINT MATRIX command would print the matrix A onto the terminal. If the size of matrix(n) is big (n>20), print the first 20 rows and columns only.

```
TRANSPOSE MATRIX
```

Syntax: TRANSPOSE MATRIX < matrix_name >

Example: TRANSPOSE MATRIX A

Function: Computes transpose of the matrix `A`. Note that `TRANSPOSE MATRIX` is an update command, which means you have to transpose matrix `IN-PLACE`.

- NOTE : Transposing a matrix IN PLACE means not using any additional disk blocks. You are allowed to use a limited amount of main memory (say 2 block). You have to transpose the matrix and write it back into the same blocks the matrix was stored in. We will keep track of blocks used while evaluating your submission, violating the block limit will result in a heavy penalty for the entire Project Phase.

```
EXPORT MATRIX
```

Syntax: EXPORT MATRIX < matrix_name >

Example: EXPORT MATRIX A

Function: Again, similar to what `EXPORT` command does for tables, `EXPORT MATRIX` command will export the matrix to the data folder as a file named `A.csv` (syntax: `<matrix_name>.csv`).

```
RENAME MATRIX
```

Syntax: RENAME MATRIX < matrix_currentname > < matrix_newname >

Example: RENAME MATRIX A AT

Function: Renames a Matrix `which is present in memory` from `A` to `AT` (from `< matrix_currentname >` to `< matrix_newname >`)

```
CHECKSYMMETRY
```

Syntax: CHECKSYMMETRY < matrix_name >

Example: CHECKSYMMETRY A

Function: Checks whether the given matrix `A` is symmetric or not. A matrix is symmetric if `A = A'`. The commands outputs `TRUE` if the matrix is symmetric. `FALSE` otherwise. You must perform this operation IN-PLACE too. The same constraint as `TRANSPOSE MATRIX` command applies for block limit.

```
COMPUTE
```

Syntax: COMPUTE < matrix_name >

Example: COMPUTE A

Function: Computes the expression `A - A'` and stores the result in `A_RESULT` (syntax: <matrix_name>_RESULT). Basically, find the transpose of the given matrix (here the transpose would be, `A'`) and subtract from the given matrix(here, `A`).

Note :

Necessary error handling like semantic, syntactic checks, whether or not the matrix etc. is loaded is expected for all commands.

Report

Submit a `Report.md` in the `docs` folder including the following details:

- Detailed Report for each command you have implemented. Mention the Logic, Page design(where applicable), number of block access based on input size, error handling etc. for each command. Each command should have a separate section titled `<Command Name>` : `COMPUTE`, `CHECKSYMMETRY`, etc.
- The Report should contain all the assumptions(if any) under the title `Assumptions`.
- Your learning and take away from this Phase, titled `Learnings`
- Contribution of each member in the Project and how you have coordinated as a Team.
