Matrix_lib

A matrix struct is declared on the matrix.h header : it contains a float** data and the size of the matrix (number of row(s) and column(s)).

To create a matrix, use the initializeMatrix(row, column) function with the different parameters. It returns a pointer to the matrix.
To free the memory used by the matrix just pass it as argument of the deleteMatrix() function.

Of course, you can make arithmetical operation, such as addition (addMatrix), substraction (subMatrix) and multiplication (mulMatrix for two matrices or mulMatScalar to multiply a matrix by a scalar number).
Each of these functions got a bool argument : if true the function will return a pointer to a matrix, wich stocks the result of the operation. If false, the result is stock on the first matrix passed as parameter and the function return NULL.

<u>Example</u> : -addMatrix(a, b, false) → the result is stock in ‘a’ matrix (first one).
-addMatrix(a, b, true) → the result is stock on a new matrix, it memory address is return by the function.

Some optionnal functions exist, such as printMatrix() properly prints a matrix function. Or the randomizeMatrix to randomize() datas of the matrix (between -1 and 1 for the moment).

sameNumOfRow(), sameNumOfColumn(), mulCompatibility() are bool function and also shortcuts for the other function. They return true or false, their name means what they do.

SUM UP :

```
struct MATRIX{
        unsigned int row;
        unsigned int column;
        float** data}; typedef struct MATRIX matrix;



//Matrix allocation & desallocation functions
matrix* initializeMatrix(unsigned int row,unsigned int column);
void deleteMatrix(matrix *m);

//Randomize matrix data between -1 & 1
void randomizeMatrix(matrix* m);

//Matrix calculus
matrix* addMatrix(matrix* a, matrix* b, bool res);
matrix* subMatrix(matrix* a, matrix* b, bool res);
matrix* mulMatrix(matrix* a, matrix* b, bool res);
matrix* mulMatScalar(matrix* a, float scalar, bool res);

//Matrix print function
void printMatrix(matrix* m);

//Matrix size check
bool sameNumOfRow(matrix* a, matrix* b);
bool sameNumOfColumn(matrix* a, matrix* b);
bool mulCompatibility(matrix* a, matrix* b);
```