```c
#include "myFunctions.h"
#include "definitions.h"
#include "params.h"
#include "typedefs.h"

void printMatrix(REAL *c, INT nrow, INT ncol)
{
#if (OUTOFF)
    INT i, j, idx;
    for (i = 0; i < nrow; i++) {
        for (j = 0; j < ncol; j++) {
            idx = j + i * ncol;
            printf("%10.2f;", c[ idx ]);
        }
        printf("\n");
    }
#endif
    printf("\n");
}

void InitializeMatrices(REAL *a, REAL *b, INT m, INT n, INT k)
{
    INT i, j, l, idx;

    // initialize matrices a & b

    for (i = 0; i < m; i++) {
        for (l = 0; l < n; l++) {
            idx      = l + i * n;
            a[ idx ] = ( REAL ) idx;
        }
    }

    for (l = 0; l < n; l++) {
        for (j = 0; j < k; j++) {
            idx      = j + l * k;
            b[ idx ] = ( REAL ) idx;
        }
    }
}

void matrixMultiply(REAL *a, REAL *b, REAL *c, INT m, INT n, INT k)
{
    INT  i, j, l;
    REAL sum = 0.f;

    // multiply the matrices C=A*B

    for (i = 0; i < m; i++) {
        for (j = 0; j < k; j++) {
            for (l = 0; l < n; l++) {
                sum += a[ l + i * n ] * b[ j + l * k ];
            }
            c[ j + i * k ] = sum;
            sum            = 0.f;
        }
    }
}

void ddot_Matrix_Mult(REAL *a, REAL *b, REAL *c, INT m, INT n, INT k)
{
    INT i, j;
    for (i = 0; i < m; i++) {
        for (j = 0; j < k; j++) {
                // calculating elements of matrix c from a &b
                //b has k stride to fetch elements from columns
            c[ j + i * k ] = cblas_ddot(n, a + i * n, 1, b + j, k);
        }
    }
```



```c
}

void daxpy_Matrix_Mult(REAL *a, REAL *b, REAL *c, INT m, INT n, INT k)
{
    INT i, j;
    for (i = 0; i < k; i++) {
        for (j = 0; j < n; j++) {
        // calculating columns of c using linear combination of
        // columns of a.a and c has n and k strides respectively
            cblas_daxpy(m, b[ i + j * k ], a + j, n, c + i, k);
        }
    }
}

void dgemm_Matrix_Mult(REAL *a, REAL *b, REAL *c, INT m, INT n, INT k)
{
    REAL alpha = 1.f;
    REAL beta  = 1.f;
 // cblas is column-major. So a is transposed to make it row major.
    cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, m, n, k, alpha, a, k,
 b, n, beta, c, n);
}
```