Experiment No. 5

NAME: Omkar Khanvilkar	ROLL NO: 07
CLASS: TY_IT-B	BATCH: 2
DATE OF PERFORMANCE: 20/08/2024	PRN No: 12210313

Problem Statement:

Implement multithreading for Matrix Operations using Pthreads.

We will implement the following Matrix Operations:

1) Matrix Addition :

The dimensions of both the matrices(operands) must be same to perform addition.

2) Matrix Subtraction

The dimensions of both the matrices(operands) must be same to perform addition.

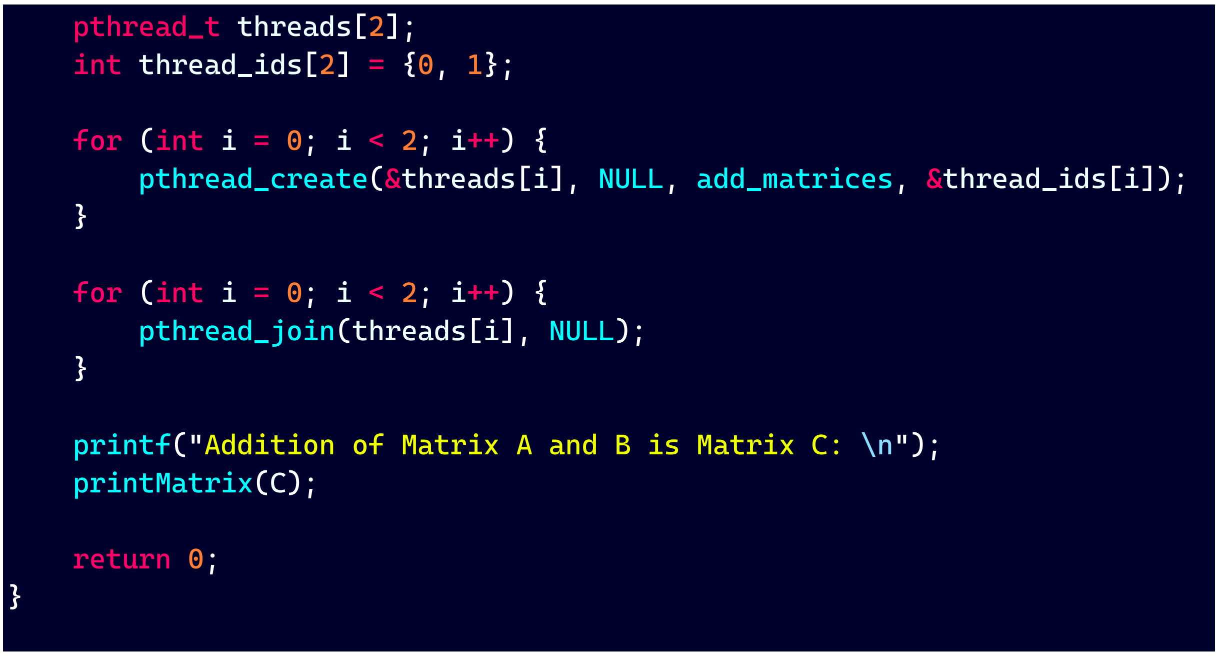
3) Matrix Multiplication

The number of columns in the first matrix must be equal to the number of rows in the second matrix to perform multiplication.

Matrix Addition

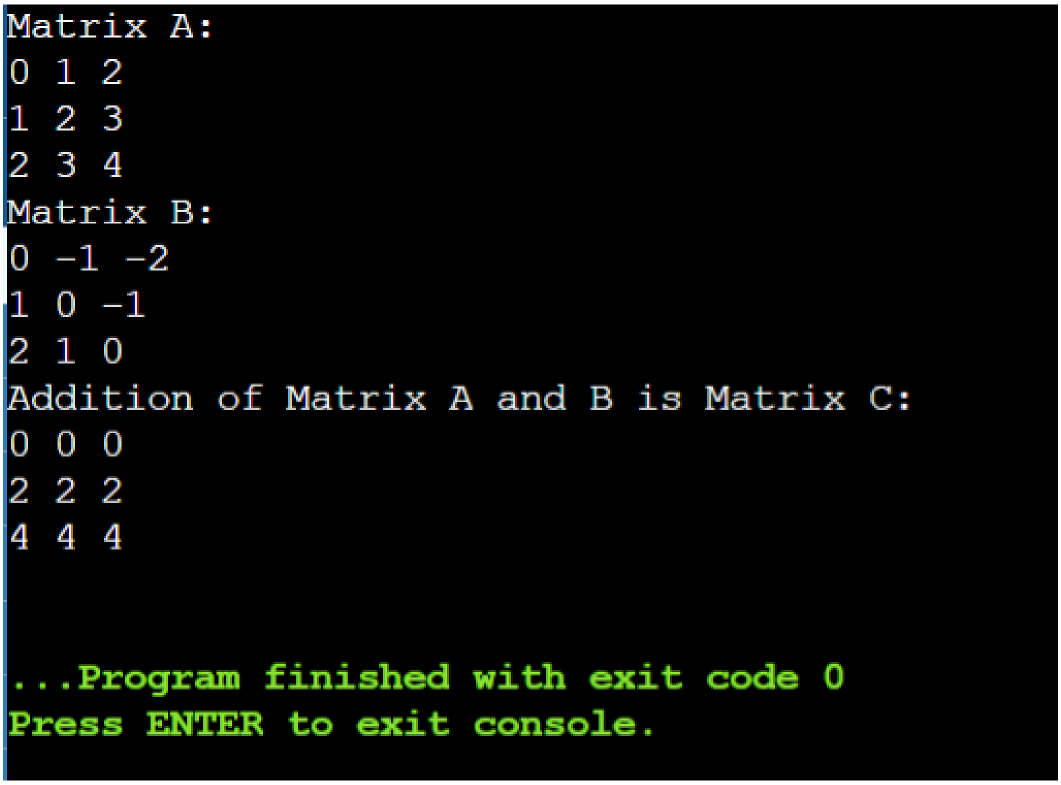
Code:

```
#include <stdio.h>
#include <pthread.h>

#define N 3

int A[N][N], B[N][N], C[N][N];

void* add_matrices(void* arg) {

    int thread_id = *(int*)arg;

    for (int i = thread_id; i < N; i += 2) {
        for (int j = 0; j < N; j++) {
            C[i][j] = A[i][j] + B[i][j];
        }
    }
    return NULL;
}

void printMatrix(int matrix[N][N]){
    for(int i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            printf("%d ",matrix[i][j]);
        }
        printf("\n");
    }
}

int main() {
    // Initialize matrices A , B and C
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            A[i][j] = i + j;
            B[i][j] = i - j;
            C[i][j] = 0;
        }
    }

    // print matrices A and B
    printf("Matrix A:\n");
    printMatrix(A);

    printf("Matrix B:\n");
    printMatrix(B);
```



```
    pthread_t threads[2];
    int thread_ids[2] = {0, 1};

    for (int i = 0; i < 2; i++) {
        pthread_create(&threads[i], NULL, add_matrices, &thread_ids[i]);
    }

    for (int i = 0; i < 2; i++) {
        pthread_join(threads[i], NULL);
    }

    printf("Addition of Matrix A and B is Matrix C: \n");
    printMatrix(C);

    return 0;
}
```

Output:



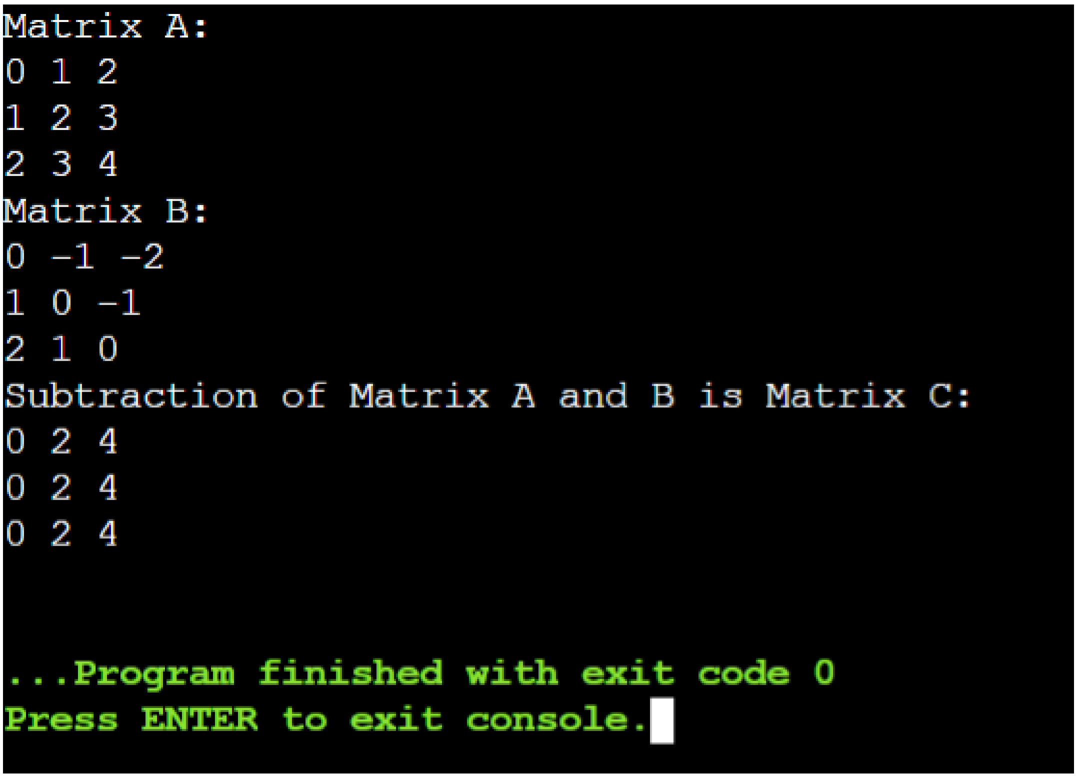
```
Matrix A:
0 1 2
1 2 3
2 3 4
Matrix B:
0 -1 -2
1 0 -1
2 1 0
Addition of Matrix A and B is Matrix C:
0 0 0
2 2 2
4 4 4


...Program finished with exit code 0
Press ENTER to exit console.
```

Matrix Subtraction

Code:

```
#include <stdio.h>
#include <pthread.h>

#define N 3

int A[N][N], B[N][N], C[N][N];

void* subtract_matrices(void* arg) {

    int thread_id = *(int*)arg;

    for (int i = thread_id; i < N; i += 2) {
        for (int j = 0; j < N; j++) {
            C[i][j] = A[i][j] - B[i][j];
        }
    }
    return NULL;
}

void printMatrix(int matrix[N][N]){
    for(int i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            printf("%d ",matrix[i][j]);
        }
        printf("\n");
    }
}

int main() {
    // Initialize matrices A , B and C
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            A[i][j] = i + j;
            B[i][j] = i - j;
            C[i][j] = 0;
        }
    }

    // print matrices A and B
    printf("Matrix A:\n");
    printMatrix(A);

    printf("Matrix B:\n");
    printMatrix(B);
    pthread_t threads[2];
    int thread_ids[2] = {0, 1};

    for (int i = 0; i < 2; i++) {
        pthread_create(&threads[i], NULL, subtract_matrices,
&thread_ids[i]);
    }

    for (int i = 0; i < 2; i++) {
        pthread_join(threads[i], NULL);
    }

    printf("Subtraction of Matrix A and B is Matrix C: \n");
    printMatrix(C);

    return 0;
}
```

Output:

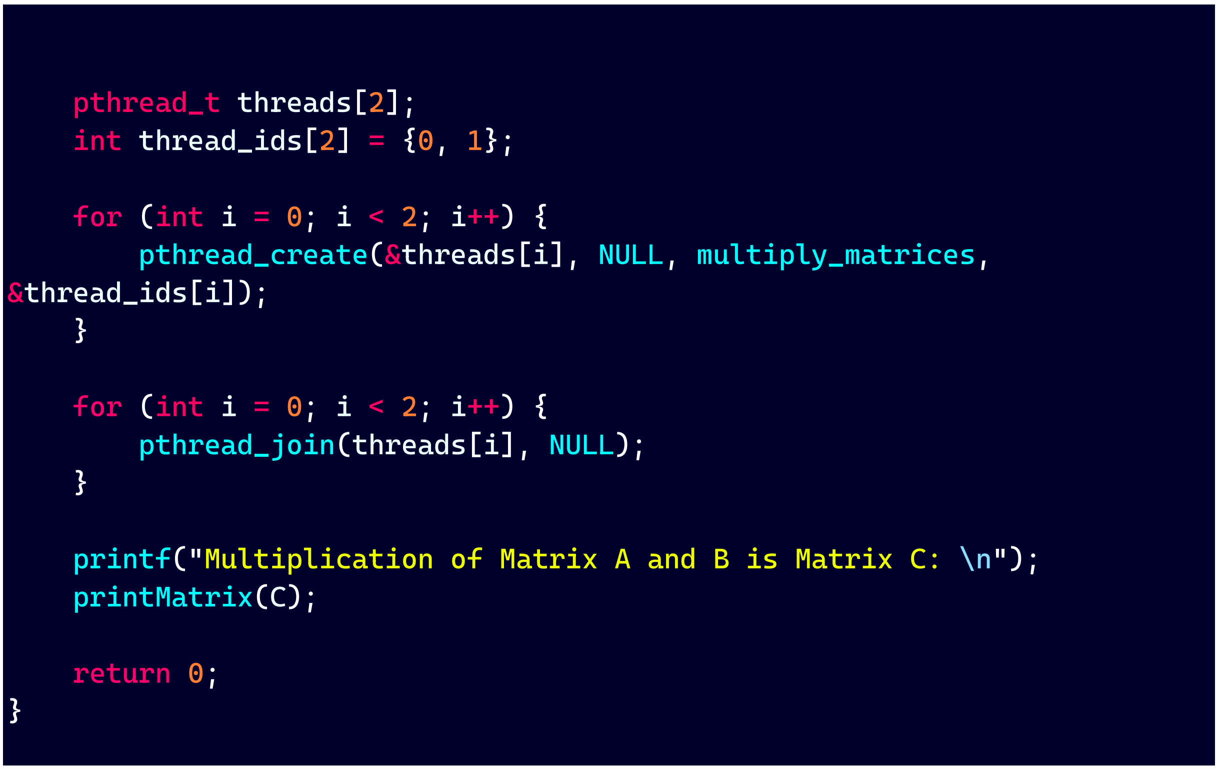


```
Matrix A:
0 1 2
1 2 3
2 3 4
Matrix B:
0 -1 -2
1 0 -1
2 1 0
Subtraction of Matrix A and B is Matrix C:
0 2 4
0 2 4
0 2 4


...Program finished with exit code 0
Press ENTER to exit console.
```

Matrix Multiplication

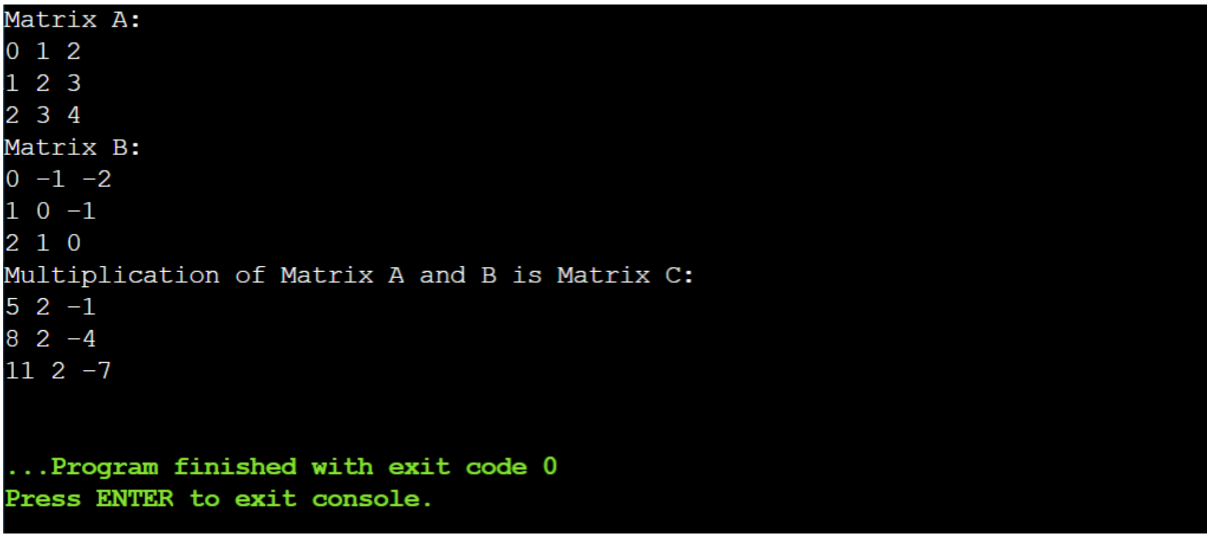
Code:

```
#include <stdio.h>
#include <pthread.h>

#define N 3

int A[N][N], B[N][N], C[N][N];

void* multiply_matrices(void* arg) {

    int thread_id = *(int*)arg;

    for (int i = thread_id; i < N; i += 2) {
        for (int j = 0; j < N; j++) {
            for(int k = 0; k < N; k++){
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return NULL;
}

void printMatrix(int matrix[N][N]){
    for(int i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            printf("%d ",matrix[i][j]);
        }
        printf("\n");
    }
}

int main() {
    // Initialize matrices A , B and C
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            A[i][j] = i + j;
            B[i][j] = i - j;
            C[i][j] = 0;
        }
    }

    // print matrices A and B
    printf("Matrix A:\n");
    printMatrix(A);

    printf("Matrix B:\n");
    printMatrix(B);
```



```
    pthread_t threads[2];
    int thread_ids[2] = {0, 1};

    for (int i = 0; i < 2; i++) {
        pthread_create(&threads[i], NULL, multiply_matrices,
&thread_ids[i]);
    }

    for (int i = 0; i < 2; i++) {
        pthread_join(threads[i], NULL);
    }

    printf("Multiplication of Matrix A and B is Matrix C: \n");
    printMatrix(C);

    return 0;
}
```

Output:



```
Matrix A:
0 1 2
1 2 3
2 3 4
Matrix B:
0 -1 -2
1 0 -1
2 1 0
Multiplication of Matrix A and B is Matrix C:
5 2 -1
8 2 -4
11 2 -7


...Program finished with exit code 0
Press ENTER to exit console.
```