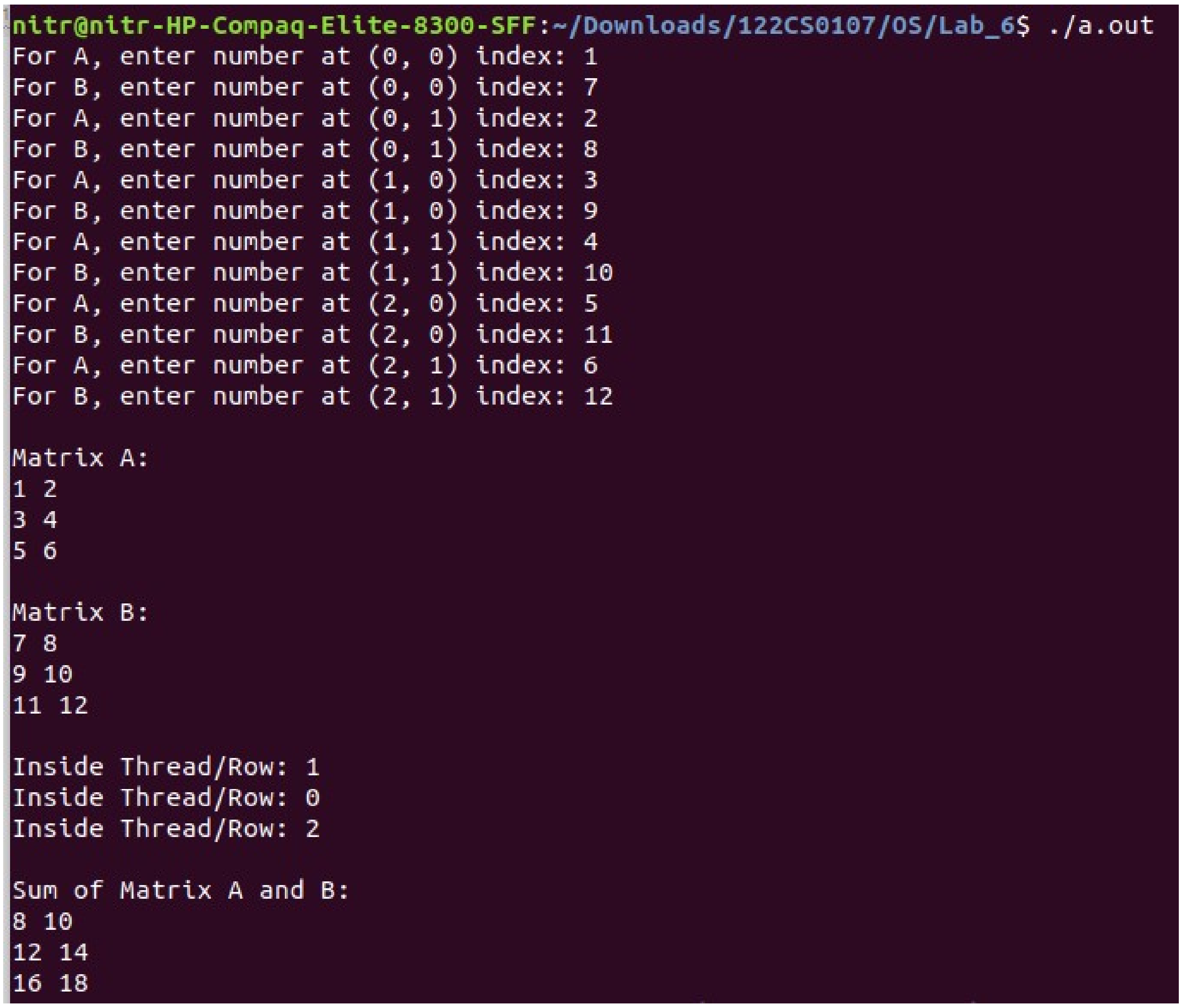
Assignment 6 | Lab 6
Operating Systems Lab
Arnav Samal – 122CS0107

<u>Question 1:</u>

Input and add two (mxn) matrices. Main thread creates m child threads. Thread1 computes the 1st row, Thread2 computes the 2nd row, ..., Threadm computes the mth row elements of the result matrix.

Code:
```
#include<stdio.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>

#define m 3
#define n 2

int A[m][n], B[m][n], SUM[m][n];
pthread_t thread[m];

void *addition(void *arg) {
        int row = (int)arg;
        printf("Inside Thread/Row: %d\n", row);
        for(int j = 0; j < n; j++) {
                SUM[row][j] = A[row][j] + B[row][j];
        }
}


int main() {
        // Input
        for (int i = 0; i < m; i++) {
                for (int j = 0; j < n; j++) {
                        int temp;
                        printf("For A, enter number at (%d, %d) index: ", i, j);
                        scanf("%d", &temp);
                        A[i][j] = temp;
                        printf("For B, enter number at (%d, %d) index: ", i, j);
                        scanf("%d", &temp);
                        B[i][j] = temp;
                }
        }


        // Displaying A
        printf("\nMatrix A:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    // Displaying B
    printf("\nMatrix B:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", B[i][j]);
        }
        printf("\n");
    }

         printf("\n");
    // Performing Addition
    int row = 0;
    // Creating Thread
    for (int i = 0; i < m; i++) {
         pthread_create(&thread[i], NULL, &addition, (int *)row);
         row++;
    }
    // Waiting for Thread to Finish
    for (int i = 0; i < m; i++) {
         pthread_join(thread[i], NULL);
    }



         // Display Addition of mA and mB
    printf("\nSum of Matrix A and B:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", SUM[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output:



```
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_6$ ./a.out
For A, enter number at (0, 0) index: 1
For B, enter number at (0, 0) index: 7
For A, enter number at (0, 1) index: 2
For B, enter number at (0, 1) index: 8
For A, enter number at (1, 0) index: 3
For B, enter number at (1, 0) index: 9
For A, enter number at (1, 1) index: 4
For B, enter number at (1, 1) index: 10
For A, enter number at (2, 0) index: 5
For B, enter number at (2, 0) index: 11
For A, enter number at (2, 1) index: 6
For B, enter number at (2, 1) index: 12

Matrix A:
1 2
3 4
5 6

Matrix B:
7 8
9 10
11 12

Inside Thread/Row: 1
Inside Thread/Row: 0
Inside Thread/Row: 2

Sum of Matrix A and B:
8 10
12 14
16 18
```

Question 2:

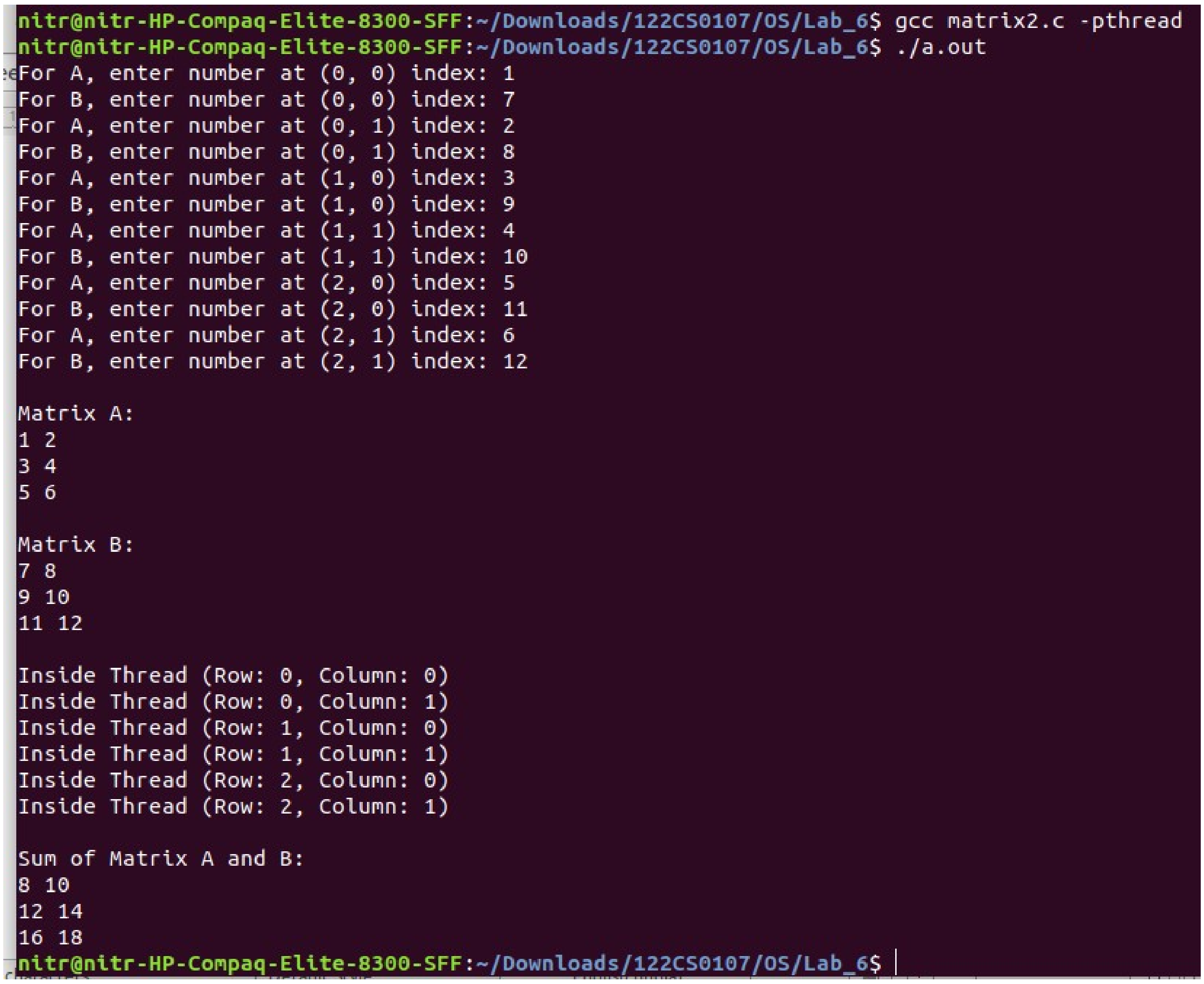
Modify the program to create m*n threads.

Code:

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

#define m 3
#define n 2

int A[m][n], B[m][n], SUM[m][n];
pthread_t thread[m][n];

struct arg_struct {
    int row;
    int column;
```

```
};

void *addition(void *args) {
    struct arg_struct *index = (struct arg_struct *)args;
    int row = index->row;
    int column = index->column;

    printf("Inside Thread (Row: %d, Column: %d)\n", row, column);
    SUM[row][column] = A[row][column] + B[row][column];
    free(args);
}

int main() {
    // Input
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("For A, enter number at (%d, %d) index: ", i, j);
            scanf("%d", &A[i][j]);
            printf("For B, enter number at (%d, %d) index: ", i, j);
            scanf("%d", &B[i][j]);
        }
    }

    // Displaying A
    printf("\nMatrix A:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    // Displaying B
    printf("\nMatrix B:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", B[i][j]);
        }
        printf("\n");
    }

    printf("\n");

    // Performing Addition
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            struct arg_struct *args = malloc(sizeof(struct arg_struct));
            args->row = i;
            args->column = j;
            pthread_create(&thread[i][j], NULL, &addition, (void *)args);
        }
    }

    // Waiting for Threads to Finish
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            pthread_join(thread[i][j], NULL);
        }
    }

    // Display Addition of A and B
    printf("\nSum of Matrix A and B:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", SUM[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output:



```
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_6$ gcc matrix2.c -pthread
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_6$ ./a.out
For A, enter number at (0, 0) index: 1
For B, enter number at (0, 0) index: 7
For A, enter number at (0, 1) index: 2
For B, enter number at (0, 1) index: 8
For A, enter number at (1, 0) index: 3
For B, enter number at (1, 0) index: 9
For A, enter number at (1, 1) index: 4
For B, enter number at (1, 1) index: 10
For A, enter number at (2, 0) index: 5
For B, enter number at (2, 0) index: 11
For A, enter number at (2, 1) index: 6
For B, enter number at (2, 1) index: 12

Matrix A:
1 2
3 4
5 6

Matrix B:
7 8
9 10
11 12

Inside Thread (Row: 0, Column: 0)
Inside Thread (Row: 0, Column: 1)
Inside Thread (Row: 1, Column: 0)
Inside Thread (Row: 1, Column: 1)
Inside Thread (Row: 2, Column: 0)
Inside Thread (Row: 2, Column: 1)

Sum of Matrix A and B:
8 10
12 14
16 18
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_6$
```