Name: Muhammad Usman
Roll no: 19P-0116

Assignment # 3

Code:

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

#define M 3
#define K 2
#define N 3


struct  Matrix {
        int A[M][K];
        int B[K][N];
        int C[M][N];
};

```

```c
void *multipyMatrix(void *ptr){
        struct Matrix *myPtr = (struct Matrix*)ptr;

        int sum = 0;
        for(int i=0; i<M; i++)//multiplication loop
        {
                for(int j=0; j<N; j++)
                {

                        for(int k=0; k<K; k++)
                        {

                                sum=sum+ (*myPtr).A[i][k] * (*myPtr).B[k][j];

                        }
                        (*myPtr).C [i][j]=sum;
                        sum=0;
                }
        }
        printf("Multiplication of A and B:\n");//printing the result Matrix (C)
        for (int i = 0; i<M ; i++)
        {
                for (int j = 0; j < N; j++)
                {
                        printf("%d \t",(*myPtr).C[i][j]);
                }
                printf("\n");
        }
}
```

```
48
49 void assignValues()//assigns values  to the structure
50 {
51         struct Matrix m;
52         int A[M][K] = {{1, 4}, {2, 5}, {3, 6}};
53         int B[K][N] = {{8, 7, 6}, {5, 4, 3}};
54         //Assigning values to Matrix A
55         for (int i = 0; i<M ; i++)
56         {
57                 for (int j = 0; j < K; j++)
58                 {
59                         m.A[i][j] = A[i][j];
60                 }
61
62         }
63         //Assigning values to Matrix B
64         for (int i = 0; i< K ; i++)
65         {
66                 for (int j = 0; j < N; j++)
67                 {
68                         m.B[i][j] = B[i][j];
69                 }
70
71         }
72         //Matrix C (this will be the result matrix)
73         for (int i = 0; i<M ; i++)
74         {
75                 for (int j = 0; j < N; j++)
76                 {
77                         m.C[i][j] = 0;
78                 }
79
80         }
```

```
81
82         //creating a pointer for object of Matrix
83         struct Matrix *ptr = &m;
84
85         pthread_t id;
86
87         //creating a thread and calling Matrix Multiplication function
88         pthread_create(&id, NULL, multipyMatrix, (void*)ptr);
89
90         //joining the main thread
91         pthread_join(id, NULL);
92
93 }
94
95 int main(){
96
97         struct Matrix m;
98
99         assignValues(m);
100
101
102         return 0;
103 }
```

Output:


usman@usman-Inspiron-3520: ~/os_lab/os_labWork
usman@usman-Inspiron-3520:~/os_lab/os_labWork$ gedit osAssign3.c
usman@usman-Inspiron-3520:~/os_lab/os_labWork$ gcc -o osAssign3.o osAssign3.c -lpthread
usman@usman-Inspiron-3520:~/os_lab/os_labWork$ ./osAssign3.o
Multiplication of A and B:
28 23 18
41 34 27
54 45 36
usman@usman-Inspiron-3520:~/os_lab/os_labWork$