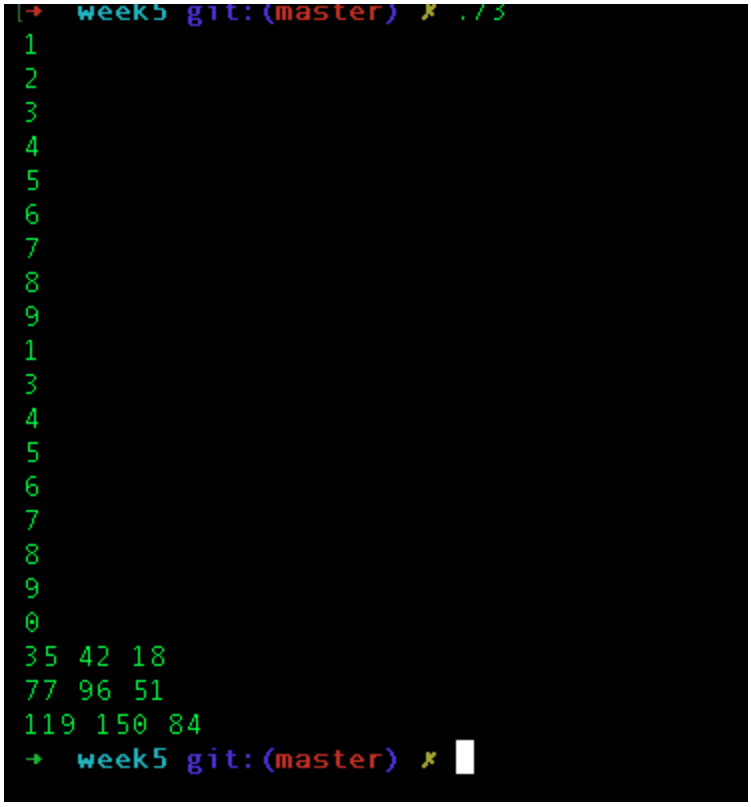
2407779_PratikDhimal
Week5

⇒ In this program we have two nested arrays. We use a nested loop to perform matrix multiplication and store the result in matrix C[N][N]. Lastly, we print the matrix using another nested loop.

⇒ In this program we have used 2 nested loops to populate the nested Arrays A[N][N] and B[N][N]. Than similar to the previous question we have used the nested loop to do necessary matrix multiplication and printing the required result



⇒ Here we have used pthread to do the same task as did in the previous question. Here we have two 3X3 matrices to be multiplied, so for this purpose we have created 9 threads so that each thread can work individually to do necessary corresponding calculations. This way race conditions will not occur.

```
[→  week5 git:(master) ✗ gcc 5.c -o 5
[→  week5 git:(master) ✗ ./5
119491
```

⇒ In this program we have a thread function which calls a counter which increases the counter to 100000. Now we create 2 threads and call the counter. We might expect to get 200000 but we get an uncertain value as two functions are accessing the same counter variable and race condition occurs.

```
[→  week5 git:(master) ✗ gcc 6.c -o 6
[→  week5 git:(master) ✗ ./6
200000
→  week5 git:(master) ✗
```

⇒ This program utilizes the **mutex** to lock the critical section so that only one thread function accesses the shared variable counter. All other concepts remain the same to the previous question.