**Parallel Computing**

**UIN: 729007151**

**Priyanka Karuppuch Samy**

Q1. The code is available in the file attached along with the zipped file.

To compile the code, I used commands

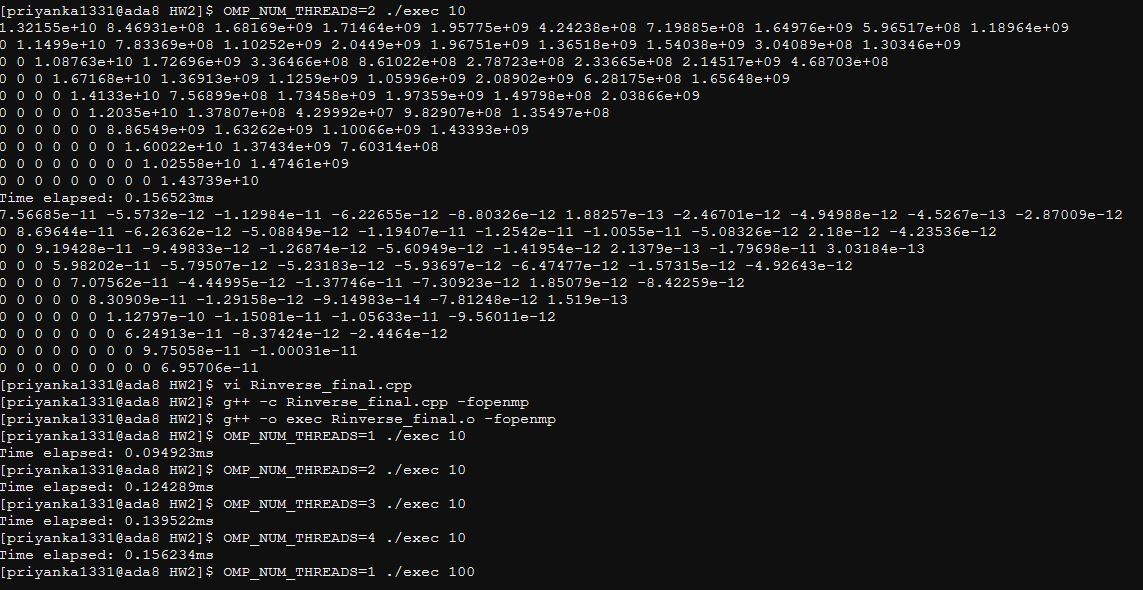
g++ -c Rinverse\_final.cpp -fopenmp

g++ -o exec Rinverse\_final.o -fopenmp

To run it, I used

CMP\_NUM\_THREADS = #n\_threads ./exec #mat\_dim

The picture below shows the execution details.



Q2) I used parallelization for recursive calls in the function that computes inverse. To improve algorithm performance, I used gauss Jordan algorithm to calculate inverse for the base conditions of recursion (i.e matrix size < 16). I created tasks for recursion operations as mentioned in the problem statement. I tried without using taskwait initially to see if the code gives correct output. But, I found out synchronization is necessary and included taskwait directive for synchronization. Apart from this, another design choice I used was to pass by reference for function implementation.

Q3) I have calculated speed up and efficiency for matrix size 900 and 2000.

Speed up and efficiency for size 900 is:

p = [1, 2, 4, 10, 20]

speed\_up = [1.0, 1.15, 1.15, 1.16, 1.15]

efficiency = [1.0, 0.57, 0.29, 0.12, 0.06]

Speed up and efficiency for size 2000 is:

p = [1, 2, 4, 10, 20]

speed\_up = [1.0, 1.13, 1.13, 1.12, 1.15]

efficiency = [1.0, 0.56, 0.28, 0.11, 0.06]

