DHRUV GARG 16 BCE 1190

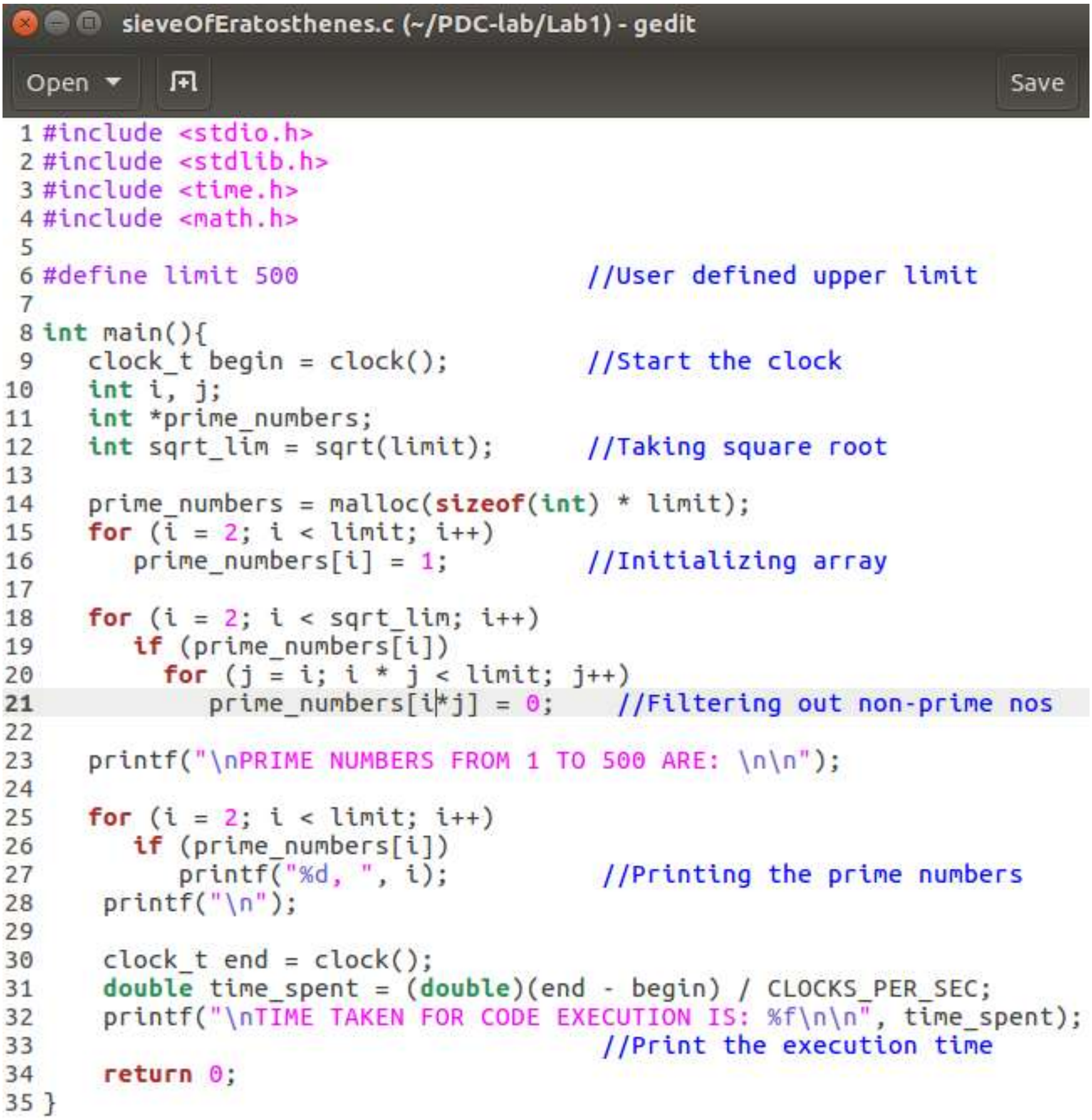
OPENMP – SIEVE OF ERATOSTHENES

LAB 1

Aim: To understand and implement firstprivate, lastprivate, parallel for and section constructs in OpenMP, for Sieve of Eratosthenes.

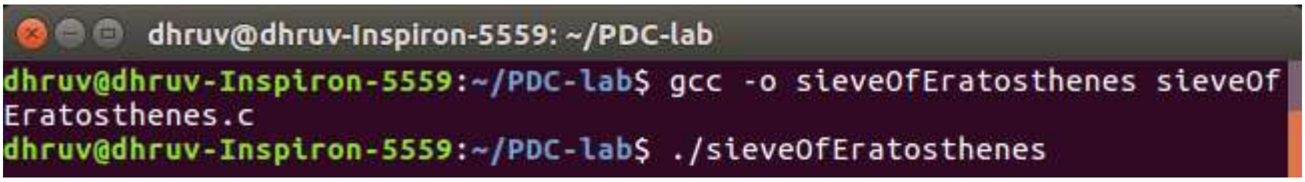
CODE AND OUTPUTS

1. Sieve of Eratosthenes – CODE WITHOUT OpenMP

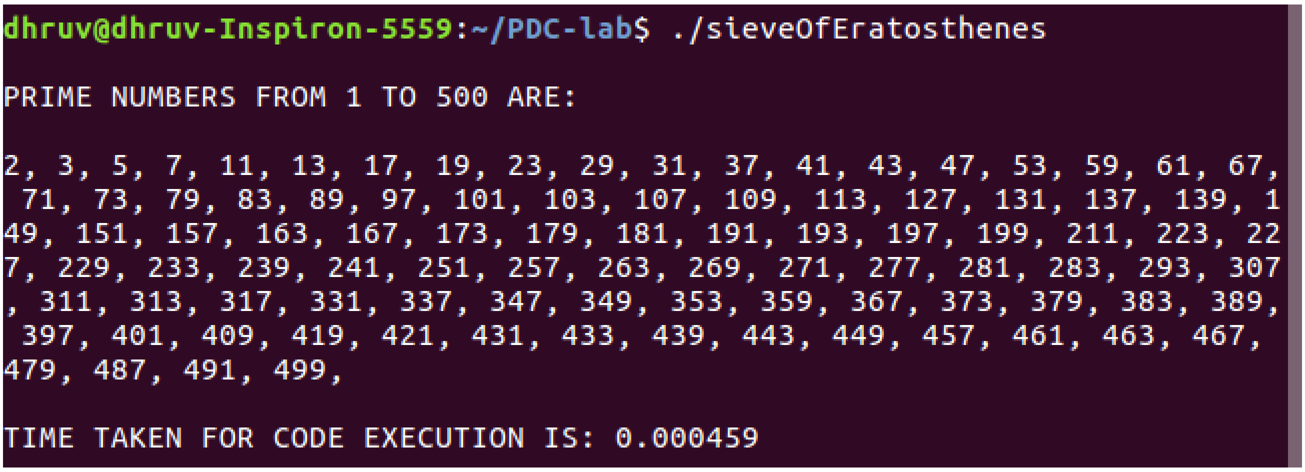


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define limit 500                   //User defined upper limit

int main(){
   clock_t begin = clock();         //Start the clock
   int i, j;
   int *prime_numbers;
   int sqrt_lim = sqrt(limit);      //Taking square root

   prime_numbers = malloc(sizeof(int) * limit);
   for (i = 2; i < limit; i++)
      prime_numbers[i] = 1;         //Initializing array

   for (i = 2; i < sqrt_lim; i++)
      if (prime_numbers[i])
        for (j = i; i * j < limit; j++)
           prime_numbers[i*j] = 0;    //Filtering out non-prime nos

   printf("\nPRIME NUMBERS FROM 1 TO 500 ARE: \n\n");

   for (i = 2; i < limit; i++)
      if (prime_numbers[i])
         printf("%d, ", i);          //Printing the prime numbers
    printf("\n");

    clock_t end = clock();
    double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
    printf("\nTIME TAKEN FOR CODE EXECUTION IS: %f\n\n", time_spent);
                                     //Print the execution time
    return 0;
}
```

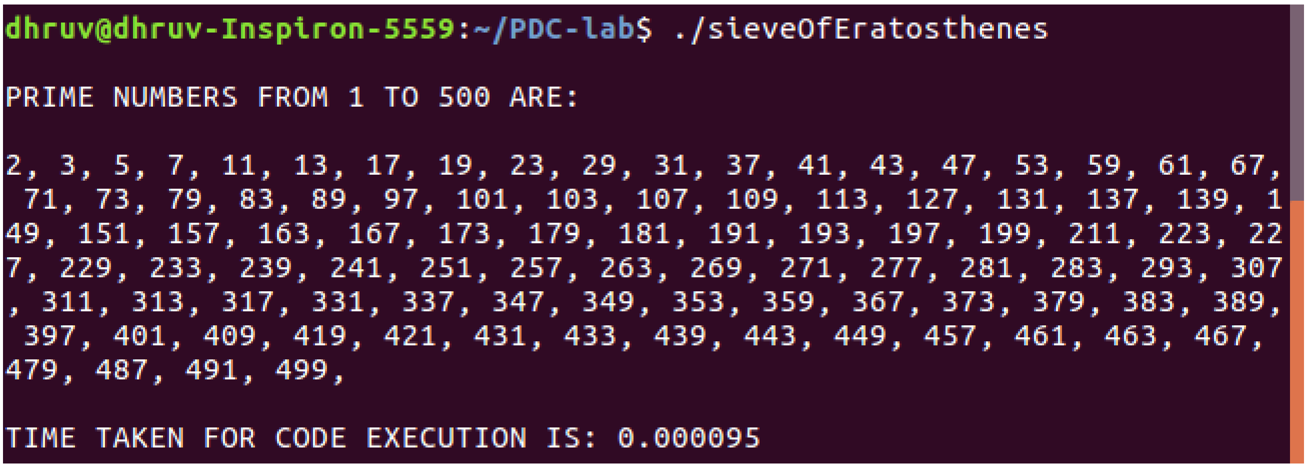
OUTPUT



```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ gcc -o sieveOfEratosthenes sieveOf
Eratosthenes.c
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./sieveOfEratosthenes
```



```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./sieveOfEratosthenes

PRIME NUMBERS FROM 1 TO 500 ARE:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 1
49, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 22
7, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307
, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467,
479, 487, 491, 499,

TIME TAKEN FOR CODE EXECUTION IS: 0.000459
```



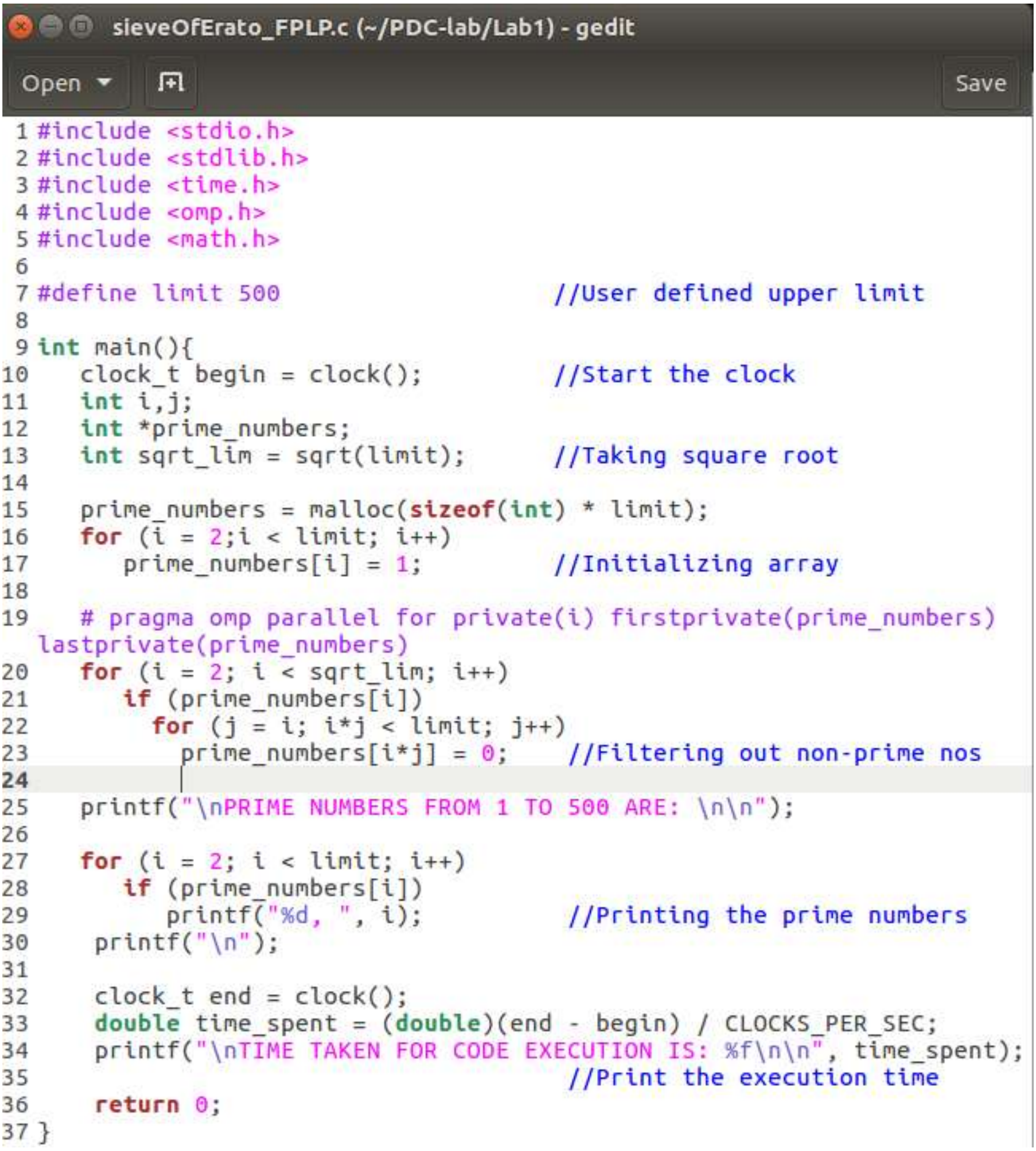
```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./sieveOfEratosthenes

PRIME NUMBERS FROM 1 TO 500 ARE:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 1
49, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 22
7, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307
, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467,
479, 487, 491, 499,

TIME TAKEN FOR CODE EXECUTION IS: 0.000095
```

INFERENCE

The inference from the above two screenshots is that although the prime numbers are found through a single thread, the execution time varies. This can be explained by the scheduling of this thread by the OS at runtime.

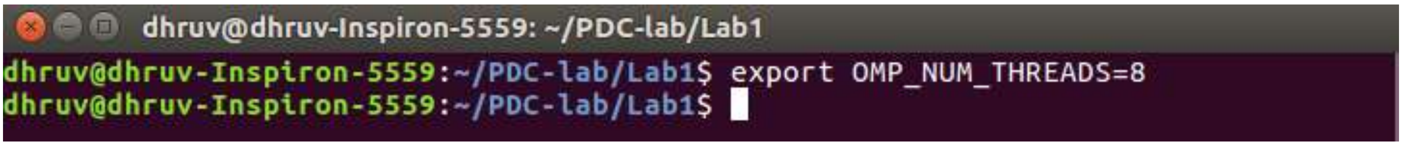
USING THE PARALLEL FOR CONSTRUCT (Continued)

2. Sieve of Eratosthenes – USING THE FirstPrivate, LastPrivate AND PARALLEL FOR CONSTRUCT

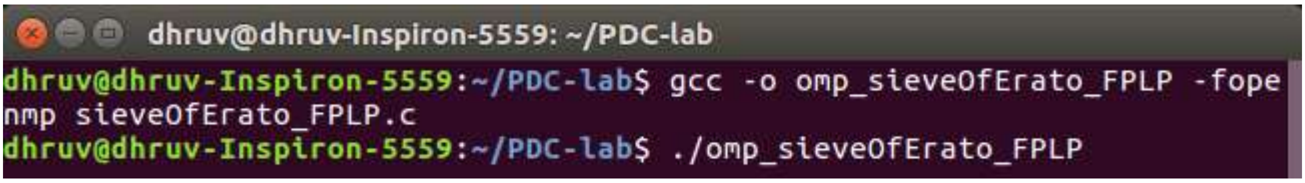


```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
#include <math.h>

#define limit 500                   //User defined upper limit

int main(){
   clock_t begin = clock();         //Start the clock
   int i,j;
   int *prime_numbers;
   int sqrt_lim = sqrt(limit);      //Taking square root

   prime_numbers = malloc(sizeof(int) * limit);
   for (i = 2;i < limit; i++)
      prime_numbers[i] = 1;         //Initializing array

   # pragma omp parallel for private(i) firstprivate(prime_numbers) lastprivate(prime_numbers)
   for (i = 2; i < sqrt_lim; i++)
      if (prime_numbers[i])
        for (j = i; i*j < limit; j++)
          prime_numbers[i*j] = 0;    //Filtering out non-prime nos

   printf("\nPRIME NUMBERS FROM 1 TO 500 ARE: \n\n");

   for (i = 2; i < limit; i++)
      if (prime_numbers[i])
         printf("%d, ", i);          //Printing the prime numbers
    printf("\n");

    clock_t end = clock();
    double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
    printf("\nTIME TAKEN FOR CODE EXECUTION IS: %f\n\n", time_spent);
                                     //Print the execution time
    return 0;
}
```

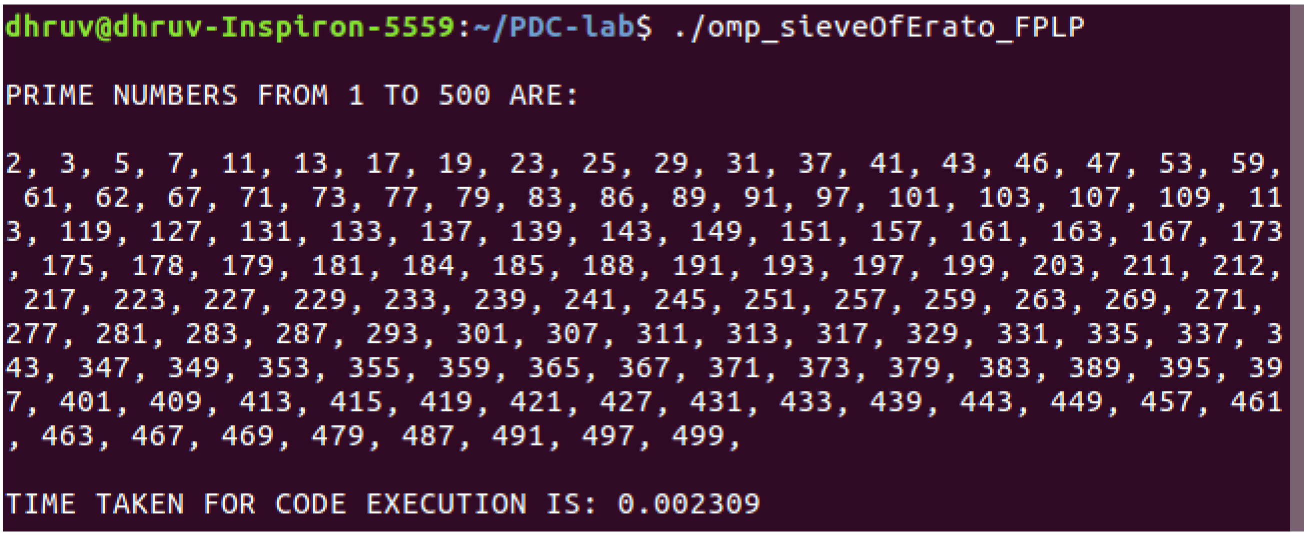
OUTPUT



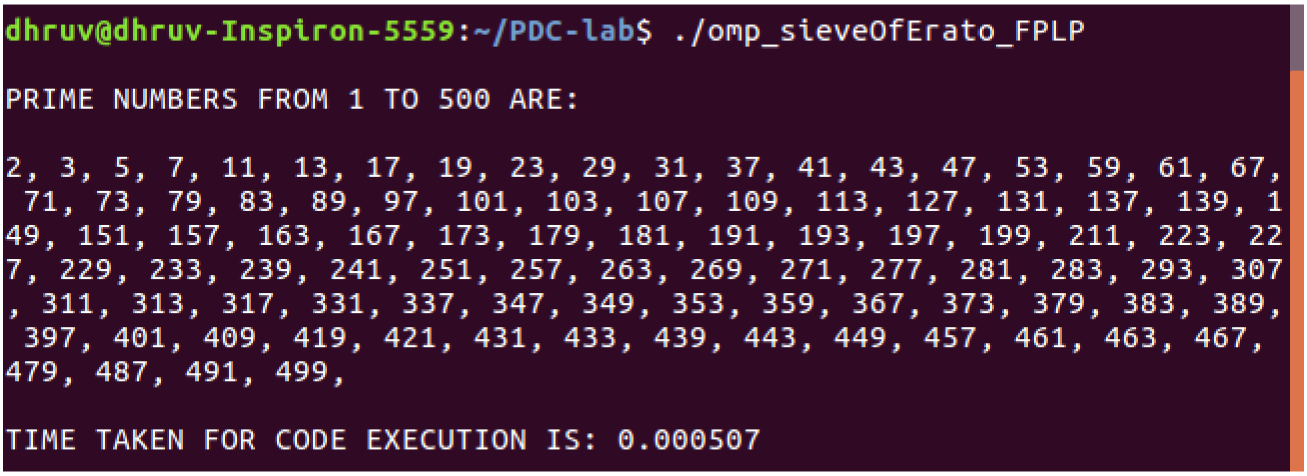
```
dhruv@dhruv-Inspiron-5559:~/PDC-lab/Lab1$ export OMP_NUM_THREADS=8
dhruv@dhruv-Inspiron-5559:~/PDC-lab/Lab1$ 
```



```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ gcc -o omp_sieveOfErato_FPLP -fope
nmp sieveOfErato_FPLP.c
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./omp_sieveOfErato_FPLP
```



```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./omp_sieveOfErato_FPLP

PRIME NUMBERS FROM 1 TO 500 ARE:

2, 3, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 37, 41, 43, 46, 47, 53, 59,
 61, 62, 67, 71, 73, 77, 79, 83, 86, 89, 91, 97, 101, 103, 107, 109, 11
3, 119, 127, 131, 133, 137, 139, 143, 149, 151, 157, 161, 163, 167, 173
, 175, 178, 179, 181, 184, 185, 188, 191, 193, 197, 199, 203, 211, 212,
 217, 223, 227, 229, 233, 239, 241, 245, 251, 257, 259, 263, 269, 271,
277, 281, 283, 287, 293, 301, 307, 311, 313, 317, 329, 331, 335, 337, 3
43, 347, 349, 353, 355, 359, 365, 367, 371, 373, 379, 383, 389, 395, 39
7, 401, 409, 413, 415, 419, 421, 427, 431, 433, 439, 443, 449, 457, 461
, 463, 467, 469, 479, 487, 491, 497, 499,

TIME TAKEN FOR CODE EXECUTION IS: 0.002309
```



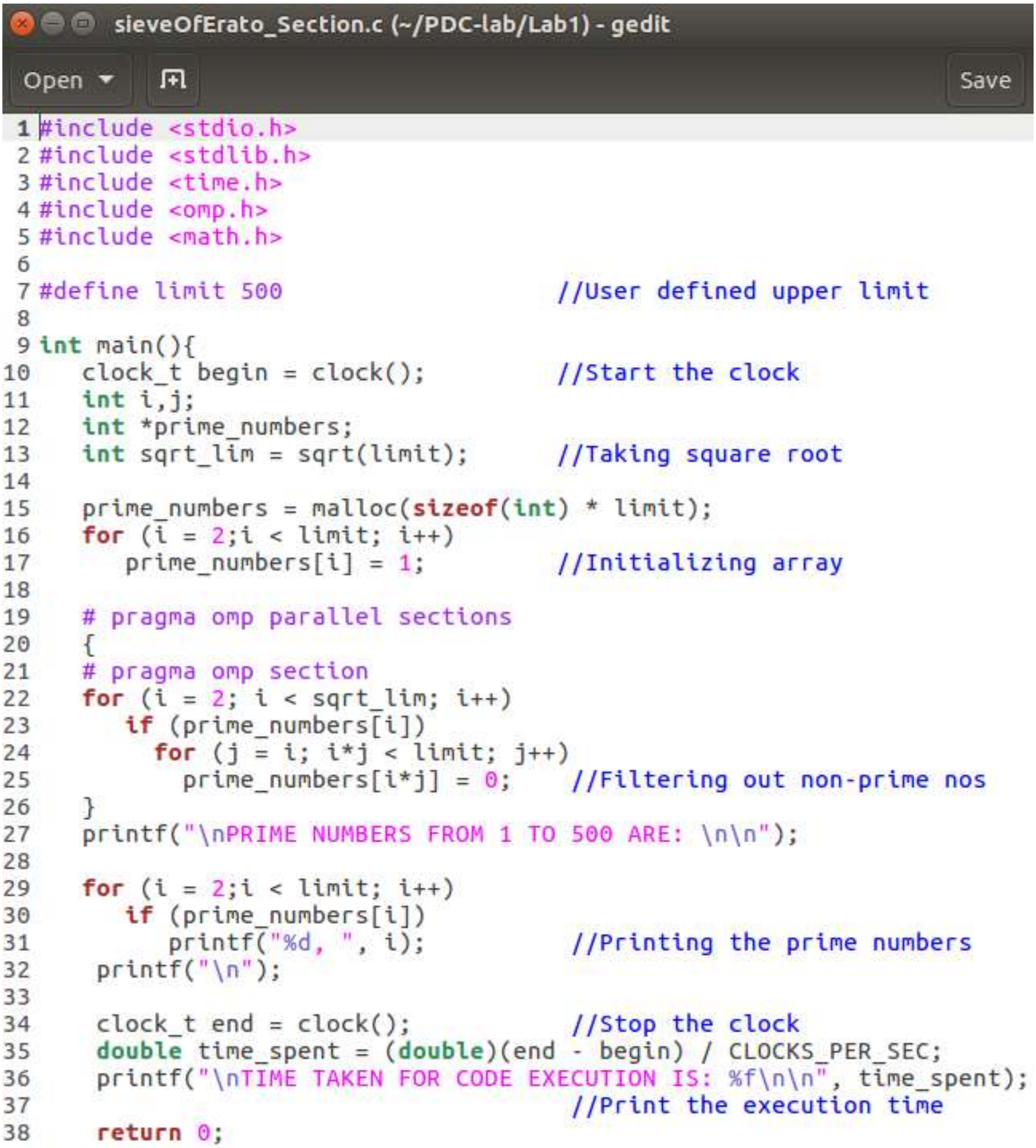
```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./omp_sieveOfErato_FPLP

PRIME NUMBERS FROM 1 TO 500 ARE:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 1
49, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 22
7, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307
, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467,
479, 487, 491, 499,

TIME TAKEN FOR CODE EXECUTION IS: 0.000507
```

INFERENCE

Number of threads used in both cases were 8. However, the execution times vary widely. This can be explained by different scheduling times of the various threads by the OS.

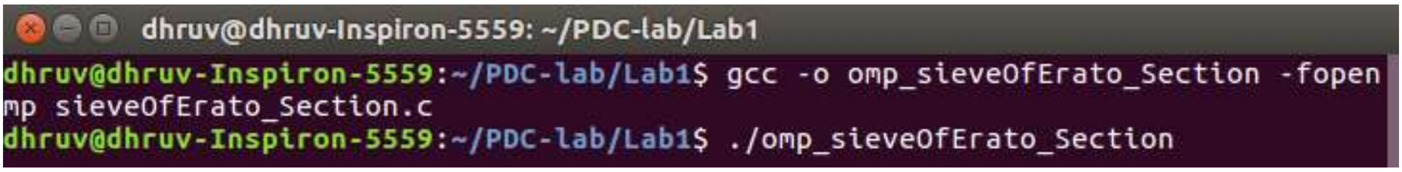
SECTIONS CONSTRUCT (Continued)

3. Sieve of Eratosthenes – USING THE SECTIONS CONSTRUCT

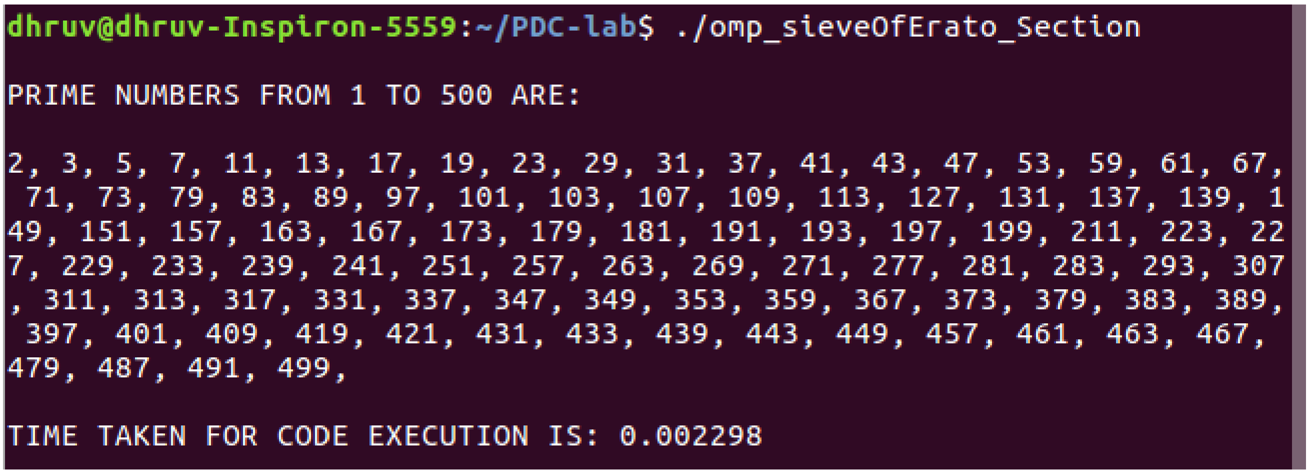


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
#include <math.h>

#define limit 500                    //User defined upper limit

int main(){
   clock_t begin = clock();          //Start the clock
   int i,j;
   int *prime_numbers;
   int sqrt_lim = sqrt(limit);       //Taking square root

   prime_numbers = malloc(sizeof(int) * limit);
   for (i = 2;i < limit; i++)
      prime_numbers[i] = 1;          //Initializing array

   # pragma omp parallel sections
   {
   # pragma omp section
   for (i = 2; i < sqrt_lim; i++)
      if (prime_numbers[i])
        for (j = i; i*j < limit; j++)
          prime_numbers[i*j] = 0;     //Filtering out non-prime nos
   }
   printf("\nPRIME NUMBERS FROM 1 TO 500 ARE: \n\n");

   for (i = 2;i < limit; i++)
      if (prime_numbers[i])
         printf("%d, ", i);           //Printing the prime numbers
    printf("\n");

    clock_t end = clock();            //Stop the clock
    double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
    printf("\nTIME TAKEN FOR CODE EXECUTION IS: %f\n\n", time_spent);
                                      //Print the execution time
    return 0;
```

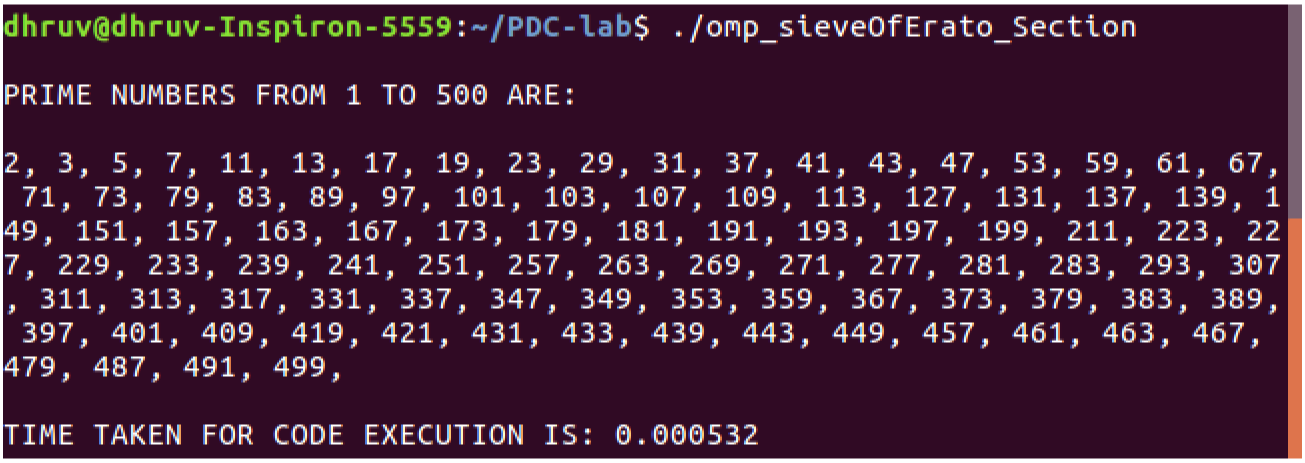
OUTPUT



```
dhruv@dhruv-Inspiron-5559:~/PDC-lab/Lab1$ gcc -o omp_sieveOfErato_Section -fopen
mp sieveOfErato_Section.c
dhruv@dhruv-Inspiron-5559:~/PDC-lab/Lab1$ ./omp_sieveOfErato_Section
```



```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./omp_sieveOfErato_Section

PRIME NUMBERS FROM 1 TO 500 ARE:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 1
49, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 22
7, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307
, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467,
479, 487, 491, 499,

TIME TAKEN FOR CODE EXECUTION IS: 0.002298
```



```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./omp_sieveOfErato_Section

PRIME NUMBERS FROM 1 TO 500 ARE:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 1
49, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 22
7, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307
, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467,
479, 487, 491, 499,

TIME TAKEN FOR CODE EXECUTION IS: 0.000532
```

INFERENCE

Number of threads used in both cases were 8. However, the execution times vary widely. This can be explained by different scheduling times of the various threads by the OS.