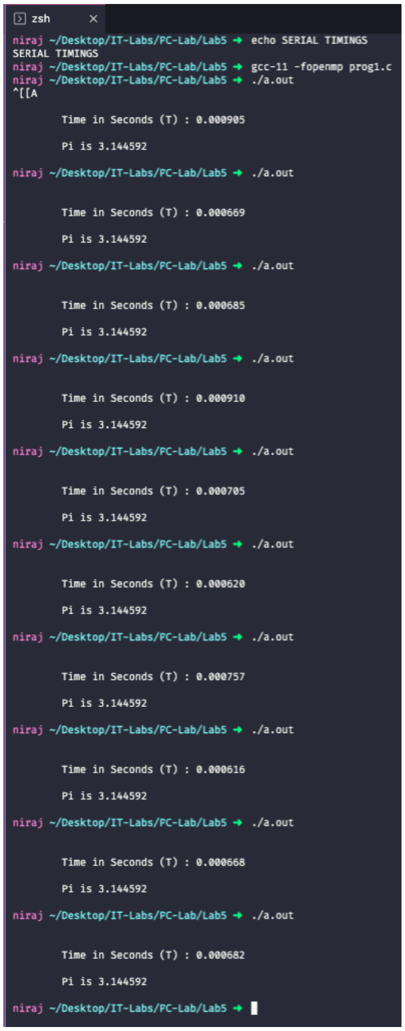
IT301 – Parallel Computing

Assignment – 5

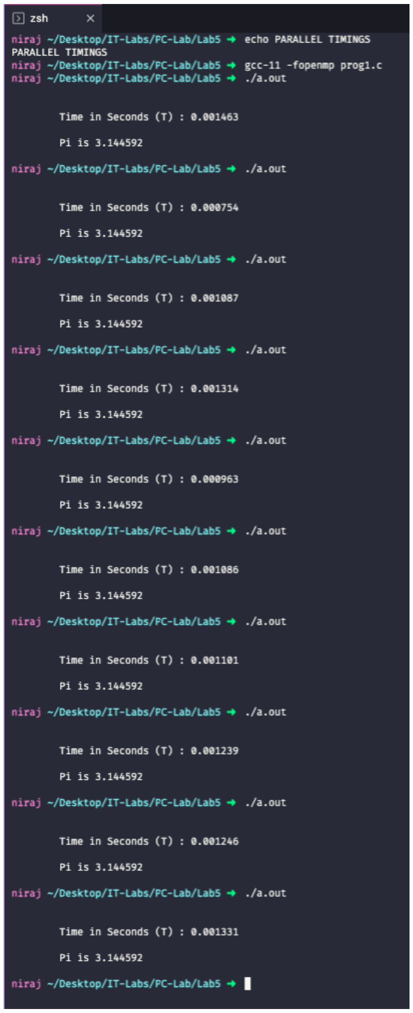
Name: Niraj Nandish

Roll No: 191IT234

1. Program 1
   a. Observation – As we can see below in the two images, the time it took for the program to run sequentially is less than the time it took for the program to run parallelly. We can say that using parallel execution for small number of iterations will be slower compared to running it sequentially.



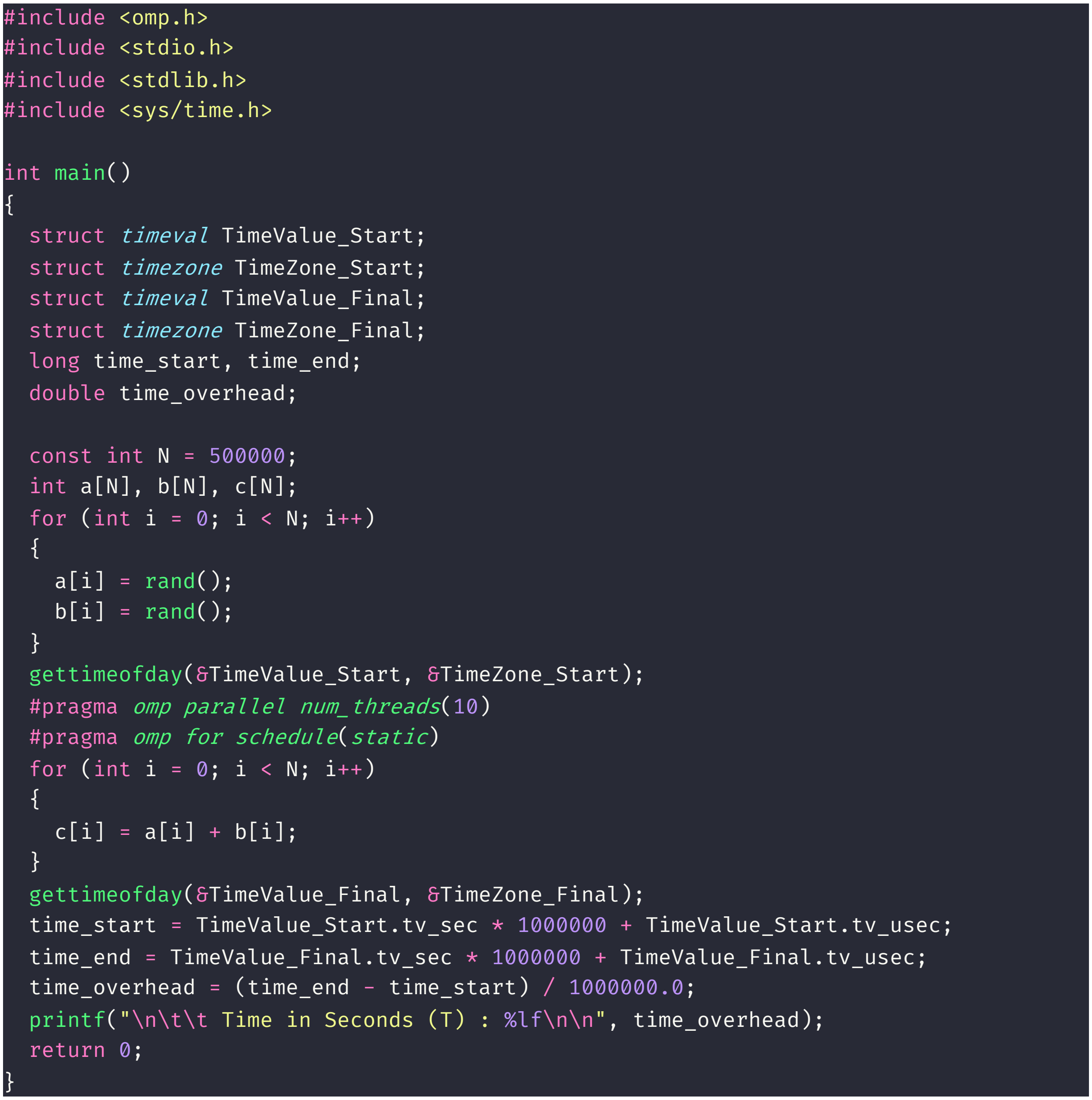
```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  echo SERIAL TIMINGS
SERIAL TIMINGS
niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  gcc-11 -fopenmp prog1.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out
^[[A

        Time in Seconds (T) : 0.000905

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.000669

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.000685

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.000910

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.000705

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.000620

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.000757

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.000616

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.000668

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.000682

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜
```



```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  echo PARALLEL TIMINGS
PARALLEL TIMINGS
niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  gcc-11 -fopenmp prog1.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.001463

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.000754

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.001087

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.001314

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.000963

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.001086

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.001101

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.001239

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.001246

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜  ./a.out


        Time in Seconds (T) : 0.001331

        Pi is 3.144592

niraj ~/Desktop/IT-Labs/PC-Lab/Lab5 ➜
```

2. Program 2

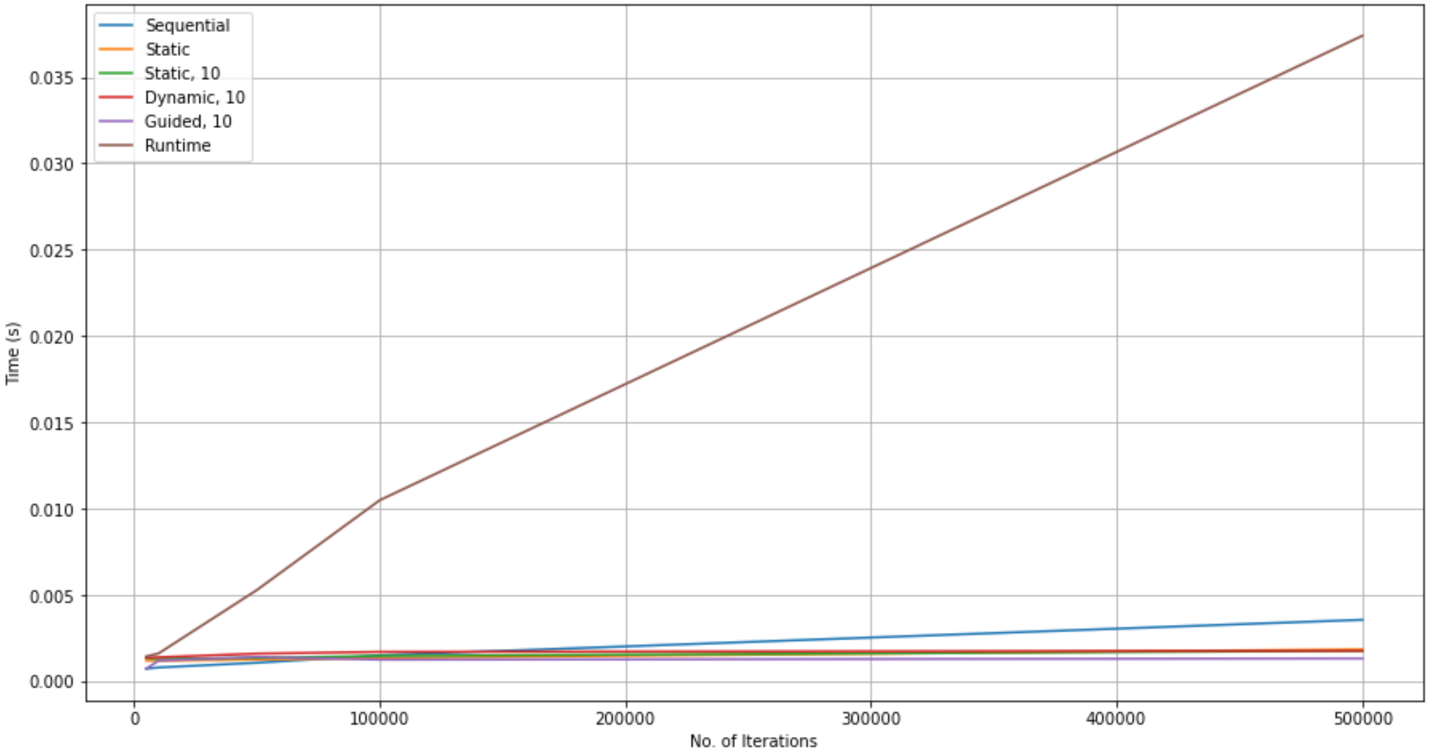
   a. Code



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

int main()
{
  struct timeval TimeValue_Start;
  struct timezone TimeZone_Start;
  struct timeval TimeValue_Final;
  struct timezone TimeZone_Final;
  long time_start, time_end;
  double time_overhead;

  const int N = 500000;
  int a[N], b[N], c[N];
  for (int i = 0; i < N; i++)
  {
    a[i] = rand();
    b[i] = rand();
  }
  gettimeofday(&TimeValue_Start, &TimeZone_Start);
  #pragma omp parallel num_threads(10)
  #pragma omp for schedule(static)
  for (int i = 0; i < N; i++)
  {
    c[i] = a[i] + b[i];
  }
  gettimeofday(&TimeValue_Final, &TimeZone_Final);
  time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
  time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
  time_overhead = (time_end - time_start) / 1000000.0;
  printf("\n\t\t Time in Seconds (T) : %lf\n\n", time_overhead);
  return 0;
}
```

b. Observation

TIME (s) → SCHEDULE () ↓	5k Iterations	10k Iterations	50k Iterations	100k Iterations	500k Iterations
Sequential	0.000762	0.000828	0.001100	0.001532	0.003587
Static	0.001229	0.001216	0.001288	0.001376	0.001877
Static, 10	0.001360	0.001318	0.001381	0.001496	0.001791
Dynamic, 10	0.001390	0.001418	0.001628	0.001732	0.001797
Guided, 10	0.000757	0.001220	0.001438	0.001280	0.001349
Runtime	0.001468	0.001640	0.005310	0.010508	0.037395



We can see from the graph that the time taken for parallel execution (except runtime) perform worse than sequential execution for lower iterations, but as the number of iterations increases it performs better. The initial worse performance is due to the overhead in creating and maintaining the threads.