NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL

DEPARTMENT OF INFORMATION TECHNOLOGY

IT 301 Parallel Computing LAB 3

1st September 2020

Faculty: Dr. Geetha V and Mrs. Tanmayee

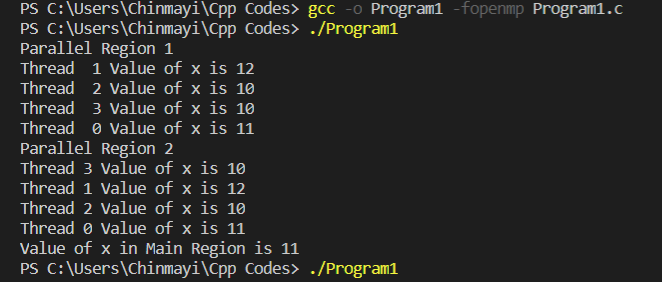
-------------------------------------------------------------------------------------------------------------------------------

**Name:** Chinmayi C. Ramakrishna

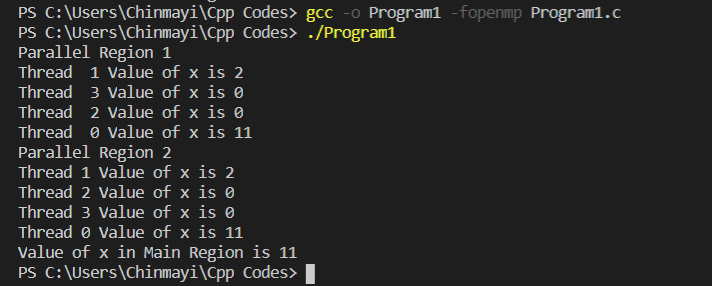
**Roll No.:** 181IT113

**Program 1:**

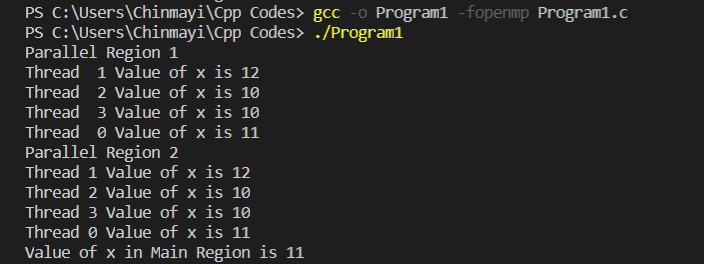
**Execute following code and observe the working of threadprivate directive and copyin clause:**



With copyin() clause. The master thread value is copied to the rest of the threads. These threads increment the initial x=10.



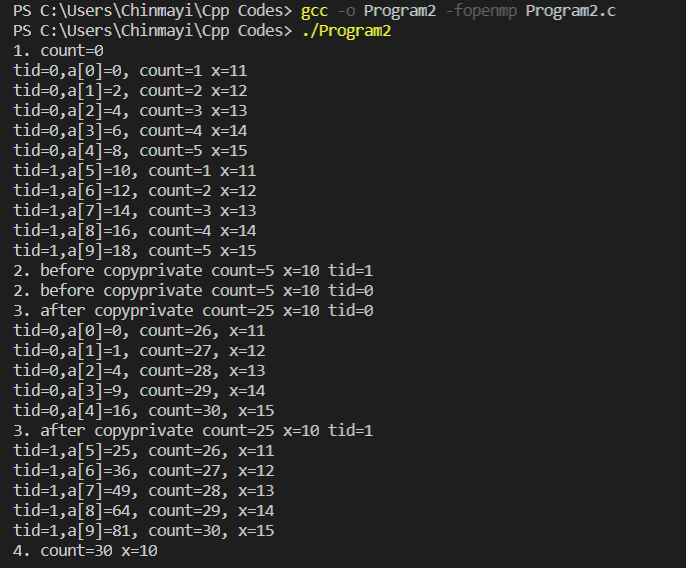
Without using copyin () clause. The master thread value is not incremented by other threads. Hence, the value of thread 1 is not 10+2 =12. Thread 0 has value 10 and rest of the threads (thread 1, thread 2, thread 3) have x value as 0.



Without using copyin () clause and initializing x globally. The output is same as using copyin() clause. By initializing x globally, all the threads have access to the same value.

**Program 2:**

**Learn the concept of firstprivate () and threadprivate ().**

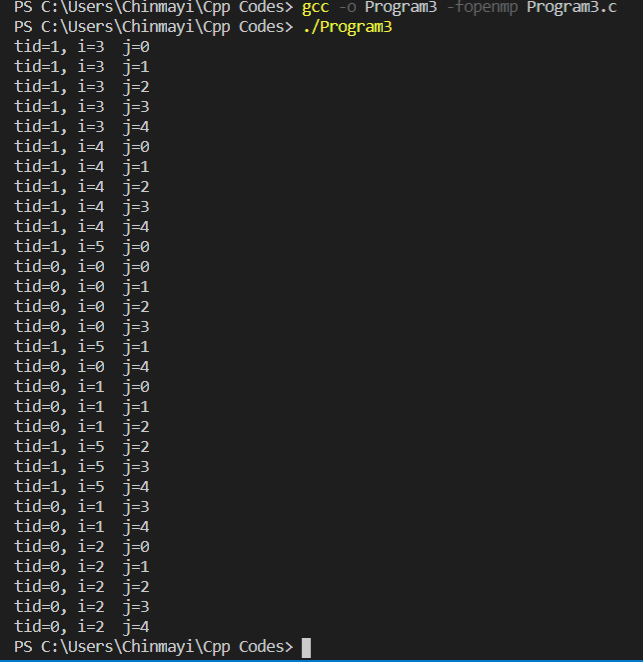


threadprivate variables are able to persist between parallel sections of the code. x value remains the same for both the threads and hence their values remain same.

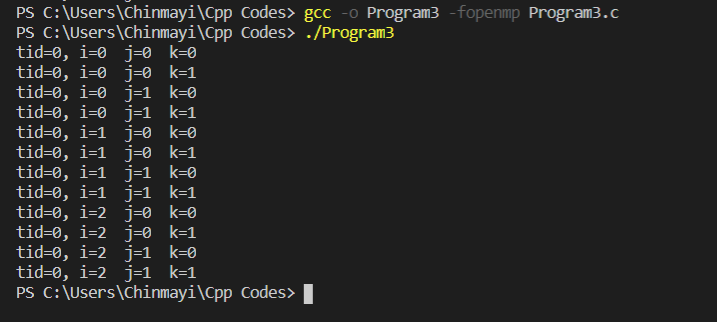
firstprivate() ensures that every thread has its own instance of the variable.

**Program 3:**

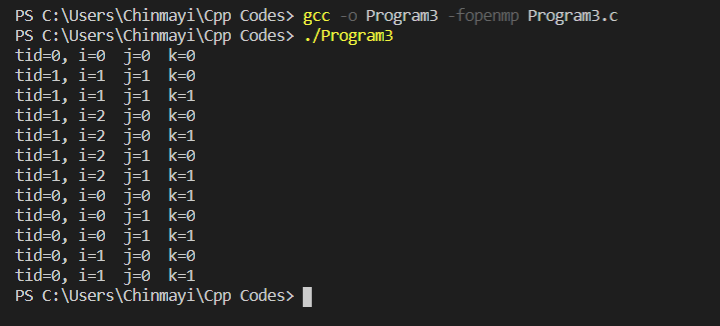
**Program to understand the concept of collapse ().**



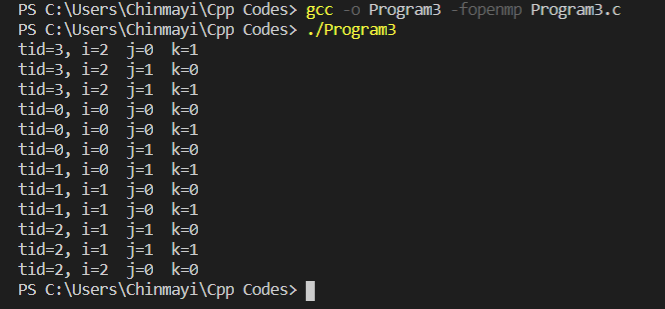
With two for loops.



Three for loops and collapse () clause. A single thread performs all the iterations.



Three for loops and collapse (2). It collapses two for loops into one large iteration space and divides it among two threads.

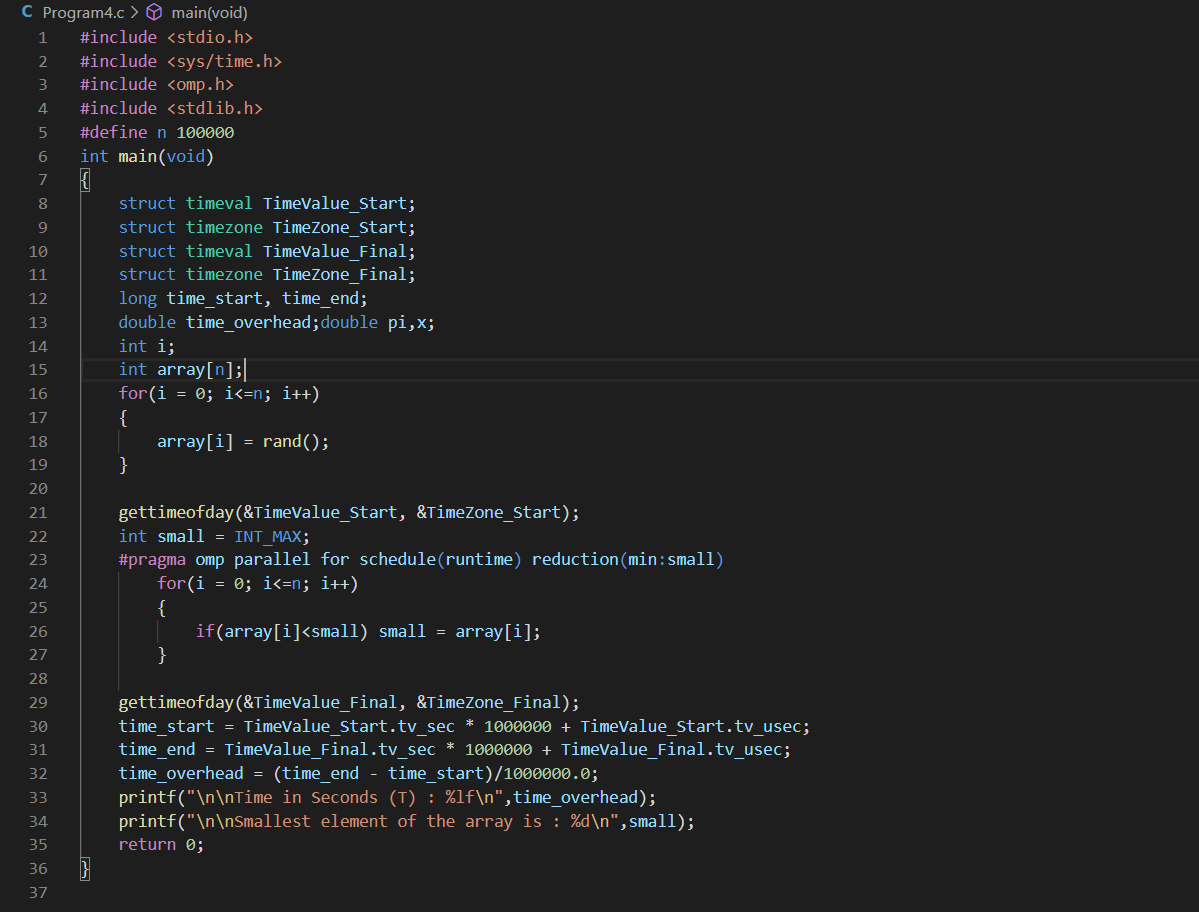


Three for loops and collapse (3). It collapses three for loops into one large iteration space and divides it among four threads.

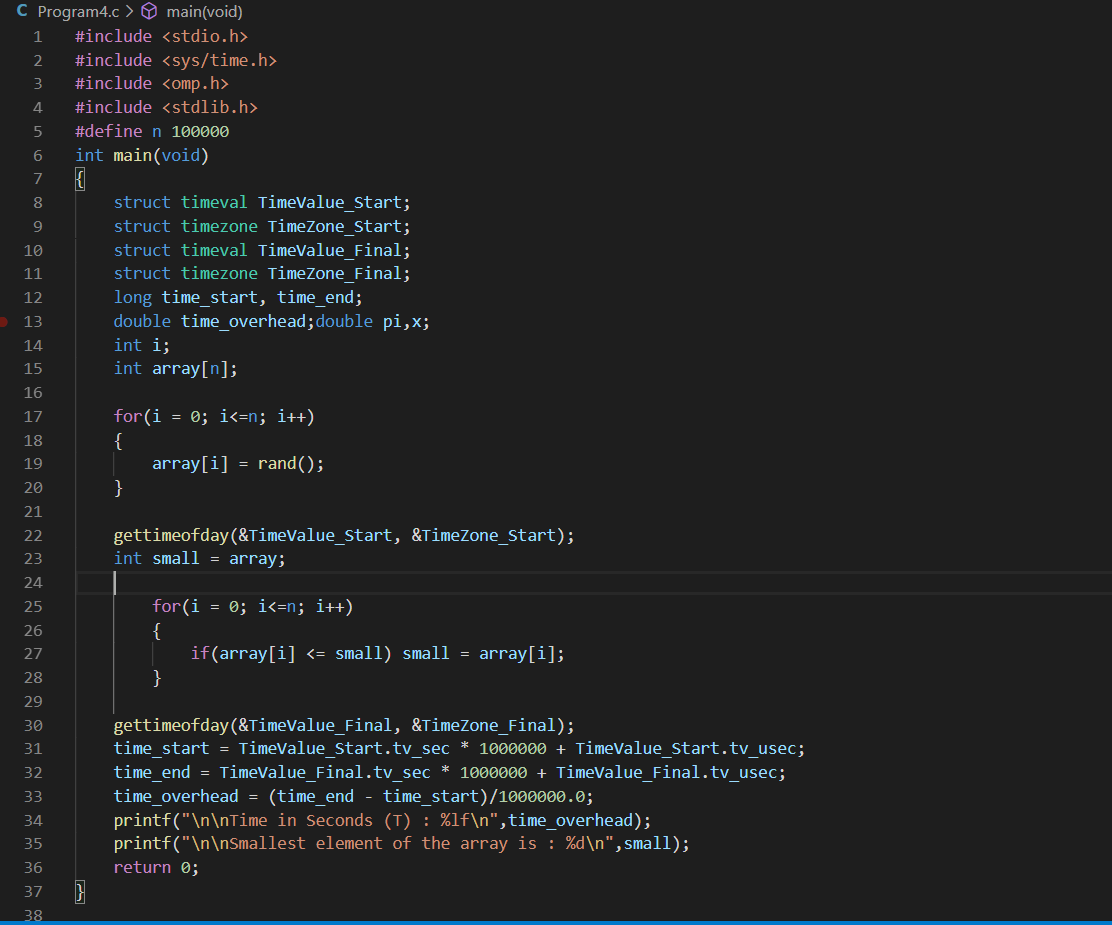
The functionality of collapse it to collapse the for loops into a single iteration. Increasing the collapse value increases parallelism as it assigns more number of threads.

**Program 4:**

**How to compare sequential and parallel program execution times.?**



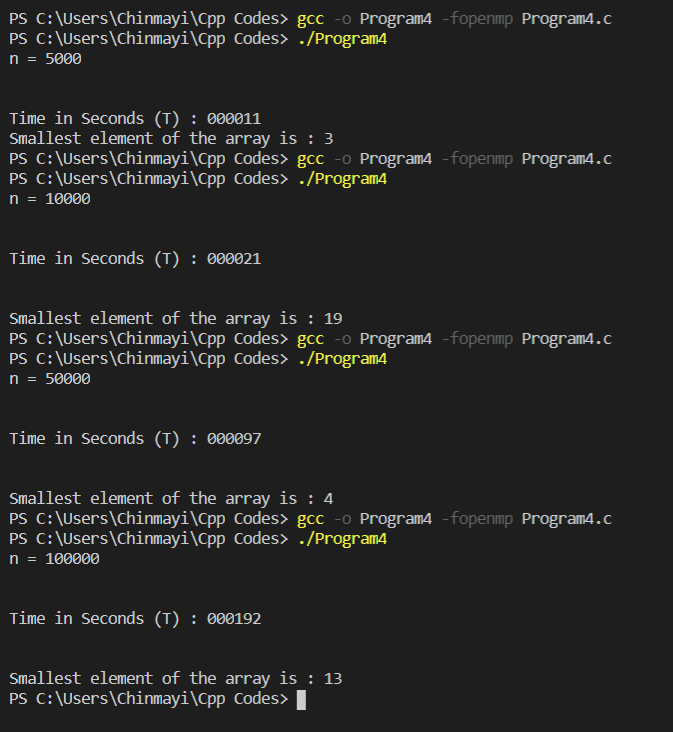
**Code for finding execution time of parallel execution for finding the smallest element in an array.**



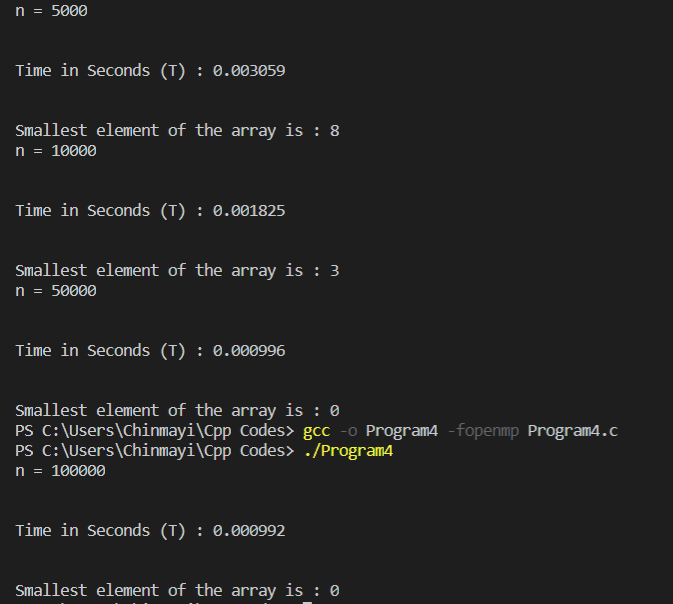
**Code for finding execution time of sequential execution for finding the smallest element in an array.**

**Outputs:**

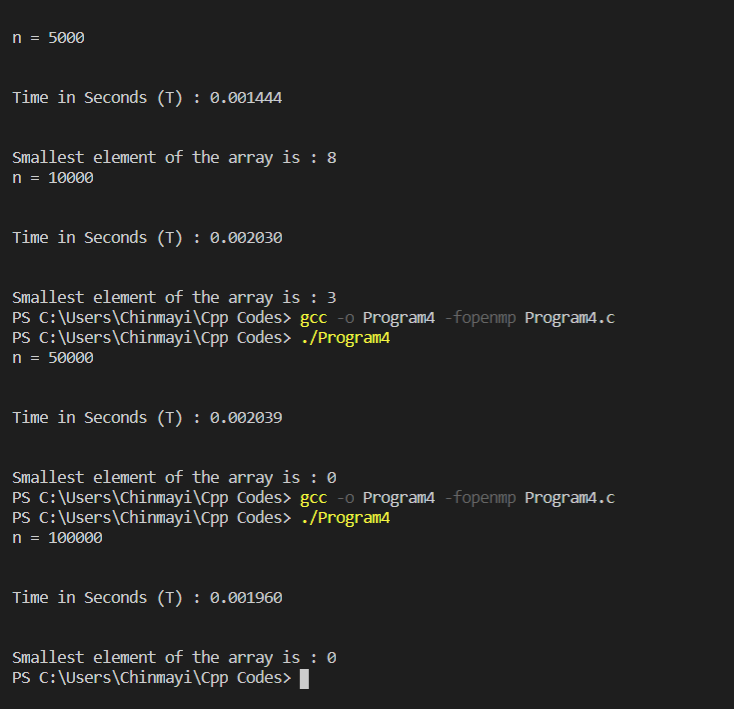
**Using sequential execution.**



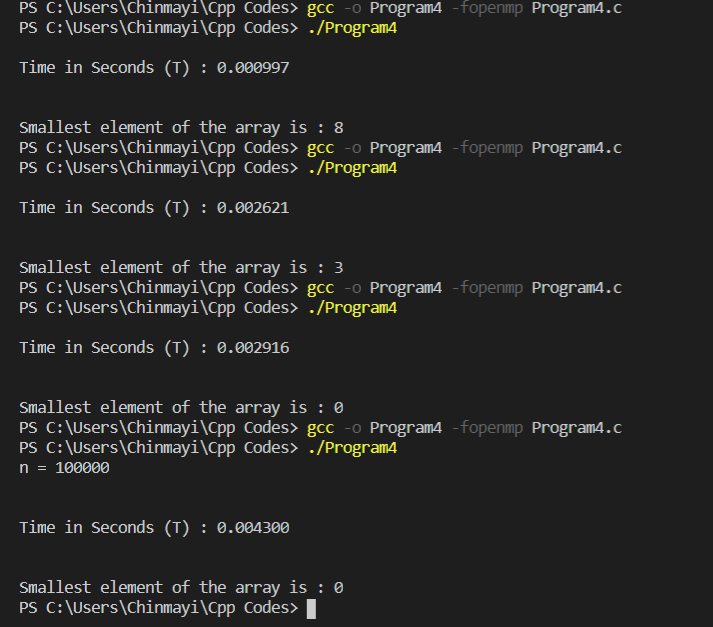
**Using schedule(static)**



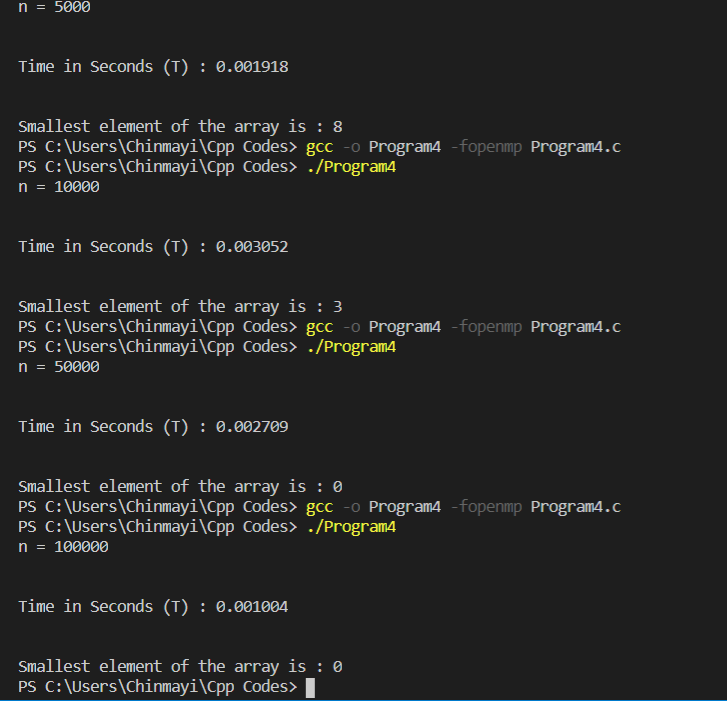
**Using schedule(static,1)**



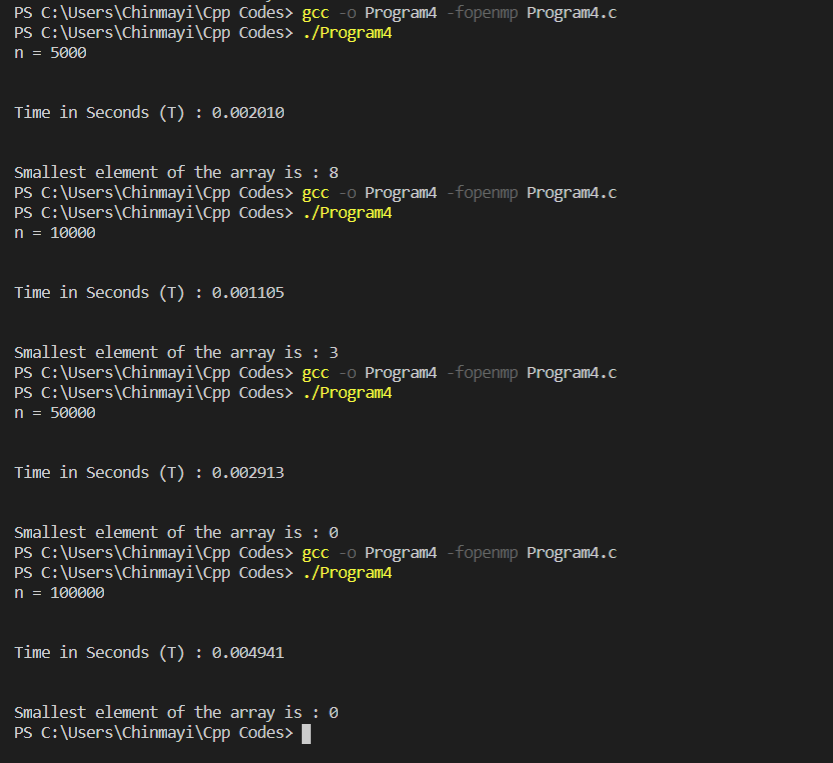
**Using schedule(dynamic, 1)**



**Using schedule(guided)**



**Using schedule(runtime)**



**At array size = 1M the parallel execution time is less than sequential execution time.**

**Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Schedule() | Total Execution time for number of iterations 5K | Total execution for number of iterations 10K | Total execution for number of iterations 50K | Total execution for number of iterations 100K |
| Sequential execution | 0.000011 s | 0.000021 s | 0.000097 s | 0.000192 s |
| static | 0.003059 s | 0.001825 s | 0.000996 s | 0.000992 s |
| Static, chunksize | 0.001444 s | 0.002030 s | 0.002039 s | 0.001960 s |
| Dynamic, chunksize | 0.000997 s | 0.002621 s | 0.002916 s | 0.004300 s |
| Guided | 0.001918 s | 0.003052 s | 0.002709 s | 0.001004 s |
| runtime | 0.002010 s | 0.001105 s | 0.002913 s | 0.004941 s |