Parallel Computing
Assignment - Pthreads
Venkata Shiva Krishna Reddy Avula
Vavula1@uncc.edu
Student id-800963515

1. Static Loop Scheduler.
    a. Please find the static_sched.cpp
    b. Please find the static_sched_plots.
    c. The results are sometimes very erratic because of the multiple threads running in calculating the total value of integral and the critical sections in the code where they all try to access the shared variable total sum. We never know if some of the threads are entering the critical section without blocking one another. So, there is a chance that iteration level speedup being almost equal to thread level or sometimes even greater than the thread level when the n is low and the intensity is also lower.
    d. When the intensity is low the time taken by the function to find the value of functions is very less as compared to the higher intensity values, and when the time taken by the function is less in calculating the values often all the threads after calculating the value of f are waiting to update the variable total sum for long times as all the threads after updating the total sum, calculate the value of function in less time and return back to the list of threads waiting to update the value of total sum. The speedup is further affected by the process of locking and unlocking the total sum variable which take considerable amount of time. This gets worse than the sequential as there is no need for waiting to update the value of total sum in sequential though the code is sequential the single thread doesn't wait for anyone else to get the access to the shared variable as in static scheduling. Here the overhead of thread creation also has a negative impact on speedup.
    e. We can observe from the graphs that as the value of n increases and the number of threads increase, the speedup is very high in thread level synchronization rather than the iteration level synchronization. This can be explained by the number of accesses to the memory location where the total is stored which is under a mutex lock. So as the n increases, the number of times the threads access the total sum variable is becomes lesser and lesser as compared to the number of times accessed in iteration level (which is constant at n). This results in parallel execution of most part of the code without waiting to change the total sum value in thread level. Finally, after all the heavier executions are done even if the threads have to wait sequentially for some cycles to update the total sum, they take very less time in being in the critical section of code. Whereas, for the iteration level, the speedup is less compared to the thread level as the threads spend far more time in the critical section of code in updating the total sum value.
2. Dynamic Loop Scheduler.
    a. Please see the dynamic_sched.cpp.
    b. Please see the dynamic_sched_plots.
    c. The speedup of chunk and thread level synchronization for 16 threads is almost the same for different intensities. The reason behind this maybe that the shared variable is always available unlocked for updating by all the threads though the number of shared variable accesses is far more in chunk than that in thread level. So, the speedup lies only in the fact

Parallel Computing
Assignment - Pthreads
Venkata Shiva Krishna Reddy Avula
Vavula1@uncc.edu
Student id-800963515

that 16 threads are executing in parallel for computing the integral sum between two values.

d. When both the values of n and intensity are large, the speedup is the highest. But, when the value of n or intensity decrease the speedup also decreases with them. When both the n and intensity are least the speedup is almost equivalent or worse than the sequential execution. The reason for the speedup being highest when the n is large and intensity if large is because this is best possible scenario for the thread level execution to efficiently work with high number of iterations and the time taken by the function being high. Vice-a-versa applies for the lesser n and intensity. With decreasing number of n, the number of iterations decreases and the overhead of thread creation also affects the speedup of the scheduler along with continuous calls to functions get next and done.