**Recitation 3: OpenMP**

Part of the solutions

**Exercise1:**

Since the variable ‘iam’ is declared outside the parallel function, it’s a shared. You can either declare it inside the function by making it a private variable, or set it as a private variable using the #pragma directive.

**Exercise2:**

Setting k and j as private variables. For each i, different threads take their own k,j.

**Exercise3:**

See the following link for solutions: <http://www.appentra.com/parallel-computation-pi/>

Yes, it’s necessary to use omp\_getwtime(). The clock() time is for recording the time in single CPU, but when you parallel your program, the program is divided and assigned to each cpu, so only record one’s time is obviously incorrect. While for the omp\_getwtime(), it records the time for certain point, and records again when the program ends. It’s like complete time for a program running.

With the 128 threads, 2 seconds speedup was achieved (serial:0.56s, OpenMP:0.35s).

clock(): *Returns the processor time consumed by the program*. So when you code is running in parallel it will add up the CPU time for each core and reports the total time.

*omp\_getwtime(): Returns a value in seconds of the time elapsed from some point*

**Exercise4:**

The most important thing in this exercise is the use of two ‘single’ directives. The first one should be added before ‘++iter; var=0.0;’. Whichever threads comes first it will add the iteration number and reset var. But we can’t add the ‘no wait’ clause since the implicit barrier in the ‘single’ directive is necessary here to avoid other threads from entering the ‘for loop’ before var or iter are updated.

Because of the barrier at the beginning of the while loop the replacement of the T mesh can happen using a nowait. From the solution code you can see, the barrier is placed right before ‘single’. This barrier cannot be replaced by the implicit barrier in the ‘single’, since in the single, some thread might update the critical variable while another thread is still in the previous iteration.

**Exercise5:**

gcc –D: Predefines name as a macro, with definition 1. The contents of the definition are tokenized and processed as if they appear during the translation phase three in a #define directive. So in our program, when DIM appears, it will be substituted by 20000, in this way, we can read the input file.

The force [][] array is shared by all threads, so updates to it should be atomic.

Without OpenMP: 0.61s for 20000 elements

With OpenMP and 128 threads and using guided schedule: 0.1s for 20000 elements