Parsa Bagheri
CIS 431: Parallel Computing
Assignment 3: OpenMP

In this assignment, Intel's API "OpenMP" is used to parallelize a 2D heat equation solver. I have used two 2D arrays of size 100X100. I've initialized the boundaries of input array with the value 100, and the inner value of -100. Then I initialized the output Array with boundaries of value 100 and left the inner values uninitialized. I have tested the run-times of OpenMP using different numbers of threads(P) and iterations(N) and the table below sums up the result of this experiment. The run-times of each of the different iterations of this experiment is mentioned in milliseconds.

	P=1	P=2	P=4	P=8	P=16
N=100	7 ms	7 ms	6 ms	8 ms	10 ms
N=1000	69 ms	65 ms	49 ms	60 ms	83 ms
N=10^4	689 ms	640 ms	472 ms	594 ms	834 ms
N=10^5	7024 ms	6440 ms	4513 ms	5856 ms	8497 ms

The system that I ran this experiment on has a 4-core, 2.8 GHz Intel Core i7 processor. If each core can support 2 threads, the best number of threads to run the program on is 8. The result of the experiment to some extent supports this hypothesis. It shows that having 16 threads execute at the same time (on this machine) results in the slowest run-time and having one thread results in the second slowest. However, unlike the assumption of this experiment, running this program on 4 threads resulted in a relatively faster performance than 8 threads.

Challenges:

In the beginning, I used "#pragma omp parallel" in the inner most for loop to parallelize only the inner loop and got the ID of each thread and assigned a chunk of the for loop to each thread. However, the results would turn out substantially slower than the serial version and inconsistencies were observed in the output of the program. Then I used "#pragma omp parallel for" for the inner for loop. After running the program 10-15 times, I didn't see any inconsistencies in the output, however, the program would still run much slower than the serial version. Then I used "#pragma omp parallel for collapse(2)" to parallelize two nested for loops. After running the program 20+ times, no inconsistencies were observed and the run time was faster than the serial version.