Using Pthreads:

To implement Pthreads, I created a new function solveNQueensThread which is used to initialize and do the opening move for the columns assigned to each thread. By doing this, I don't have to worry about shared memory because each thread has its own heap memory and dynamically allocates the memory for it's own boards. To handle the work division, I integer divide the size of the board by the number of threads and do a modular division of the same. The integer division result is the base number of columns (placing the first queen in each of the columns) and the modular division is the extra columns that have to be allocated.

Using OpenMP:

Did a similar approach using the pragma for loop and did the same initialization of the first row of the board so each thread gets a unique first board.

N Queens (for N = 15)						
	Sequential	1 thread	2 threads	3 threads	4 threads	OpenMP (4)
Time	44.20	44.91	24.22	16.75	14.27	13.45s

Part 4:

As expected, the time to execute the problem decreased by adding more threads, the surprising part is the omp implementation was faster than the pthreads implementation using 4 threads and a size 15 board.