Sum Quads

Suppose an NxN matrix is evenly split into four quadrants. Each quadrant is an N/2xN/2 matrix in size. N is an even integer. Find a N/2xN/2 matrix containing the sum, element wise, of the four quadrants.

Example:
Given the file

```
8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
```

The four matrices to add are

```
1 2 3 4        5 6 7 8
1 2 3 4        5 6 7 8
1 2 3 4        5 6 7 8
1 2 3 4        5 6 7 8

1 2 3 4        5 6 7 8
1 2 3 4        5 6 7 8
1 2 3 4        5 6 7 8
1 2 3 4        5 6 7 8
```

The resulting sum matrix is

```
12 16 20 24
12 16 20 24
12 16 20 24
12 16 20 24
```

Save the result matrix in a file that has the same format as the original.

Requirements:
1) Program must follow the same documentation rules given in the other programming assignments.
2) Program must be modular and function length must be kept to a minimum and perform/focus on one task.
3) All arrays and matrices must be dynamic.
4) While not required, a serial version can be present which should provide a solution that is correct.
5) The solution matrix is created using a cuda kernel.
6) Use command line arguments. One input file, one output file. Both input and output should have the same format, the first line of the file is the matrix size n, the remaining n lines contain n integers each.
7) Error check the command line arguments and file operations.
8) Error check cuda commands using the technique given in last class (or similar error checking that include the 4 items mentioned in some fashion).
9) Time three items:

- Overall time – start/stop a timer in the main function. This time will include reading from the file, setting up dynamic array, dumping the result, and clean up (free memory and closing files).
- Time to setup call the cuda kernel and cleanup afterwards. This is pretty much a timing of all cuda related function calls and a call to your cuda kernel.
- Time just the call to your cuda kernel.