ICS632 Assignment - Sequential Warmup

For this assignment turn in an archive of a single directory with your code in source files as specified in the exercise(s). Provide a report that answers specific questions as a PDF file named `README.pdf` with both text and graphs (no points awarded for prettiness, only for readability and content).

Run your code on a **dedicated node of the Cray** (i.e., 20 cores on 1 node). Running the programs takes a bit of time. It's not a good idea to start running code right before the due date as you may not get your results in time (e.g., due to queue waiting time).

Of course, it is recommended to write scripts/makefiles/whatever to automate running the code and gathering performance numbers.

Exercise #1: Tiling for locality [25 pts]

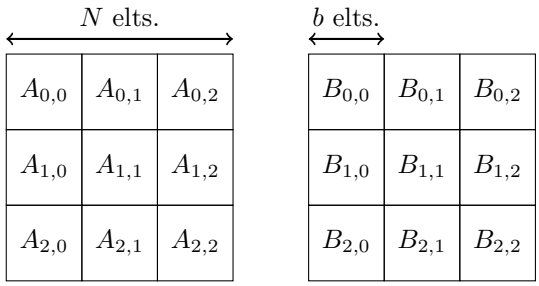
The order of for loops matters for locality when cruising through array data structures. In the case of the basic matrix multiplication implementation it is well-known that the `j` loop should be the inner loop. But, in some cases, reordering loops doesn't help. Consider an "add/transpose" code:

```
double A[N][N];
double B[N][N];

main() {
  int i,j;

  for (i=0; i < N; i++)
    for (j=0; j < N; j++)
      A[i][j] += B[j][i];
}
```

Clearly the `i-j` order is "good" for array `A` but "bad" for array `B`. Swapping the two loops would simply reverse the situation, meaning that neither order achieves good locality for both arrays.

One solution to the above conundrum is *tiling* (also called *blocking*): as opposed to cruising through entire rows/columns of the arrays, one should instead operate on (square) tiles of these arrays. For an array of size $N \times N$ and a tile size b (in number of elements), there are $(N/b) \times (N/b)$ tiles, and $b \times b$ elements in each tile, assuming that b divides N. The algorithm is rewritten as outer loops on the tile indices (from 0 to $\lceil N/b \rceil - 1$ along the vertical and horizontal dimensions), and then inner loops on each tile. Tiling maximizes data reuse

within cache lines by allowing data in loaded cache lines to be (hopefully) fully used before they are evicted.



Questions:

1. **[10 pts]** In a source file named `exercise1.c` implement both the basic algorithm above and the tiled version (for `double` arrays). The tiled version should work for **any tile size** (i.e., b may not divide N, in which case the tiles on the last tile row and the last tile column could be smaller than $b \times b$ and possibly non-square). Determine a value of N that's large enough so that the basic version runs in roughly 2 seconds when compiled with the highest optimization level (e.g., -O4, -Ofast). For this exercise always use this value of N and this optimization level.

   - Your arrays should be declared in the data segment (global variables) and not on the stack!
   - It is likely a good idea to use the C preprocessor and the `-D` compiler flag to define values for N and b at compile-time, as well as to determine at compile time whether the code runs the basic algorithm or the tiled algorithm.
   - Use the `perf` command-line tool to determine wall-clock time.

2. **[10 pts]** Using 10 trials for each tile size, execute your tiled version for all tile sizes between 1 and 300. Plot the average elapsed time vs. the tile size. Explain the trend that you observe. What is the best tile size? What speedup is achieved w.r.t. the basic version?

3. **[5 pts]** Using the `perf` command-line tool, also compute for each tile the average number of cache misses for the L1 cache (`-e L1-dcache-load-misses`) and for the LLC cache (`-e LLC-load-misses`). We only care about load misses because we never have a write of a datum that isn't just after a load of that datum. How do the number of cache misses compare between the fastest tiled version and the naive version? Add the corresponding curves to the plot from the previous question. (You may want to normalize curves in some fashion.) Discuss the trends you observe in the plot (do they make sense? if so why? is not why not?). Based on your results, would you say that it's difficult to pick a decent tile size?