CSci 532
Fall, 2020
Programming Assignment #3
Due: Dec. 1, 2020
50 points (+ bonus points)

OpenMP Matrix Transpose with Written Report

Part 1
Write a C++ program to transpose one matrix into another matrix. The program should read the numbers of rows and columns from the command line. A third but optional command line argument indicates whether or not to print the transposed matrix. The program should print the final transposed matrix if the third command line argument is a “Y” or “y”. If the print flag argument is omitted or if the flag is not “Y” or “y”, the transposed matrix should not be printed. Whether or not the print flag is present, your program should print the elapsed time to perform the transpose.

Ex.
```
$ ./transpose <rows> <cols> [<print flag>]
$ ./transpose 10 10 y
```

Your program should dynamically allocate space for each matrix based on the first two command line arguments. The matrices should both be declared to store integers. Make sure that the dimensions of the two matrices complement each other. If the original matrix is 2 rows by 3 columns, the transpose matrix should be 3 rows by 2 columns.

Part 2
Parallelize the outer loop of the transpose operation using OpenMP. Add pragma statement(s) to the source code and compile with the -fopenmp (or -qopenmp). Confirm that your transpose operation is still correct.

Print to the console the difference between the current time before and after the matrix transpose. Use the omp_get_wtime() function to collect the timing data. This represents the runtime performance of the transpose and should include any OpenMP overhead for creating and scheduling the team of threads. Be sure that your timing results do not include processing command line arguments, allocating memory, initializing matrices, or printing the transposed matrix.

Part 3
Collect data and report the runtime performance of your transpose operation. Your report must include the following details of your operating environment. This should include information about your processor (model, clock frequency), size and type of RAM, size of last level cache, and number of cores. Your operating environment also includes your operating system and version, compiler and version, and OpenMP specification level.

You should collect the runtime performance of five instances of your transpose operation with nine different configurations involving matrix size and team size. The matrix sizes should be 100x100, 1000x1000, and 2000x5000. The team sizes should be 1, 2, and 4.
(That's 5 runs of each configuration= 45 runs)

You may wish to put the timing of the transpose function in a loop and collect five data points with one run of your program. If you do this, consider whether you should iterate six times and record only runs 2 through 6.

For each configuration your report must include the individual transpose runtimes and the average runtime. For each matrix size, you should also compute the speedup for team sizes 2 and 4.

Part 4 (Bonus. Optional)
Collect and report the runtime data above using three optimization levels. Be sure to delete your executable file and recompile your program each time you change the optimization level.
(ex. make clean; make)
Optimization: -O0, -O1, -O2
That's 3x45 = 135 runs