Assignment 1: Optimized Matrix Transposition (15 Absolute Marks)

Deadline: `Sunday 15th Nov 23:00`

Submission:

Submit by email `RollNo.Zip` containing `csim.c` , `trans.c` , `parallel-trans.c` , `parallel-trans.txt` (output containing 25 time durations and speedup as gathered on a dual core, hyper-threaded machine or a quad core machine), and `parallel-trans.jpg` (the plot).

Description:

This assignment is based upon this assignment (http://reed.cs.depaul.edu/lperkovic/csc407/homeworks/cachelab.pdf) and the accompanying handout (https://staff.fnwi.uva.nl/a.visser/education/CS/perf/cachelab-handout.tar). Here is a link to :recall cache properties (http://www.ccs.neu.edu/course/com3200/parent/NOTES/cache-basics.html). Avoid finding any solutions online. They will be easily caught and besides, its a great learning exercise and you will gain nothing by looking at a solution.

There are three equal parts of this assignment (5 absolute marks each). You will only be marked on correctness. There are no style marks.

1. **Part A:** The same as in the above link.
2. **Part B:** Same as in the above link except that you need to make it work for only square matrices. Effectively, you can add (`if (M!=N) trans(M, N, A, B);` i.e. delegate to the unoptimized implementation for non-square matrices). This means you will not work on the 61x67 matrix.
3. **Part C:** Write a set of three functions `parallel_trans_odd_even` , `parallel_trans_2` , and `parallel_trans_4` in a new file `parallel_trans.c` that makes threads to divide the task of transposing. M and N will be multiple of 8.

```
`void parallel_trans_odd_even(int M, int N, int A[N][M], int B[M][N]); // one thread handles odd rows and another handles even rows`
`void parallel_trans_2(int M, int N, int A[N][M], int B[M][N]); // your best optimization for 2 threads`
`void parallel_trans_4(int M, int N, int A[N][M], int B[M][N]); // your best optimization for 4 threads`
```

Also write a new `main` in `test-parallel-trans.c` that calls each of these three functions and the basic trans and optimized trans directly. You need to measure the time for each call. You should repeat the call 1000 times and divide the time by 1000 to get a good reading. You'll have 5 data points. Repeat this experiment for 16x16, 32x32, 64x64, 128x128, and 256x256 matrices. Your program should output these 25 times along with the speedup of each time compared to the time taken by non-parallel unoptimized trans for that size. So the speedup of non-parallel unoptimized trans for each size is 1x.

Plot these 40 speedup points on a graph with the x-axis listing the 8 experiments, the y-axis showing speedup and all 5 points of each one size connected with a line. So the legend will use five colors (one for each matrix size) and the plot will have 5 lines, one of each color. You can generate the plot in Excel, Google Sheets, etc.