HW3 Due Date Wed Oct 22 7am

Pthreads:

Sort.
You have a very big file that need to be sorted alphabetically. Implement the sorting with three threads as follows:

1. Reader thread - given a large text file as input, read 1000 lines at a time. Pass each block of 1000 lines to the sort thread. The last block may contain less than 1000 lines.

2. Sort thread - sort each block of 1000 lines, write the results to a temporary file.

3. Merge thread - whenever there is more than one temporary file, open two of them and merge them into a third temporary file. Delete the two inputs.

The reader thread will need to coordinate with the sort thread to hand off each block. A blocking queue would work well. It will also have to coordinate with both the sort and merge threads to tell them the total number of blocks, so they will know when to finish. The reader thread won't know that number until it reaches the end of the file. If there are N blocks, then the sort thread will process N blocks, and the Merge thread will do N-1 merges.

OpenMP:

1. Implement parallel back substitution with OpenMP
   Recall that when we solve a large linear system, we often use Gaussian elimination followed by backward substitution. Gaussian elimination converts an nxn linear system into an upper triangular linear system by using row operations:
   - Add a multiple of one row to another row
   - Swap two rows
   - Multiply one row by a non-zero constant

   An upper triangular system has zeros below the diagonal extending from the upper left-hand corner to the lower right-hand corner
   $2x-3y=3$
   $4x-5y+z=7$
   $2x-y-3z=5$
   Can be reduced to:
   $2x-3y=3$
   $y+z=1$
   $-5z=0$

Now this system can be easily solved by first finding z using last equation, then finding y using second equation, and finally finding x using the first equation.

We can devise a couple a serial algorithms for back substitution. A*x=b

The row oriented version:

```
for(row=n-1; row>=0;row--){
   x[row]=b[row];
   for(col=row+1;col<n; col++){
            X[row]-=A[row][col]*x[col];
   x[row] /=A[row][row];

}
```

Colum oriented:

```
 for(row=0; row<n;row++)
   x[row]=b[row];
 for(col=n-1;col>=0; col--){
            X[col] /=A[col][col];
              For (row=0;row<col; row++)
                        x[row] -= A[row][col]*x[col];

}
```

a. Can the outer loop of the row implementation be parallelized in OpenMP?
b. Can the inner loop of the row implementation be parallelized in OpenMP?
c. Can the outer loop of the column implementation be parallelized in OpenMP?
d. Can the inner loop of the column implementation be parallelized in OpenMP?
e. Write OpenMP for the loops that can be parallelized
f. Modify the parallel loop with a schedule clause and thest the program with various schedules. Which schedule has better performance? Assume your system has 10000 variables