AMSC/CMSC 662 Homework 4 , Fall 2013
Due: 9:30am Thursday, October 24.

1. (10 points) **Message passing programming.** (At least, a special case of it.) Write a function `mpAllSolve.c` that uses a master and k worker threads with message passing.

```
void mpAllSolve(double (*f)(double),
                double a, double b, double L, double tol,
                double* Roots, double* PossibleRoots,
                int sizeRoots, int* nroots, int* npossroots,
                int k)
```

There will be $k+1$ queues of incoming messages, one for each thread. Each message contains an integer and two doubles.

Thread 0 maintains a stack of unprocessed subintervals, initially having one entry: `(a, b, fa, fb)`. It also maintains a length $k+1$ array of integers, 0 if thread j is idle and 1 if thread j is busy, initialized to zeros.

It loops until the stack is empty and the threads are all idle.

At each iteration,

- It checks for incoming messages and uses them to update the stack and the arrays of roots and possible roots.
- Then it assigns one entry in the stack (if there are any entries) to each idle thread (if there are any idle threads). To do this, it pops `a, b, fa, fb` off the stack and sends two messages with content `(1, a, b)` and `(2, fa, fb)`.

After exiting the loop, it sends a shut-down message to each thread, with the contents `(0, 0.0, 0.0)`, and then returns to the calling program.

Thread j ($j = 1, \ldots, k$) loops forever. At each iteration,

- It waits until it receives a message.
- If it is the shut-down message, it terminates.
- Otherwise it receives a second message and processes the interval as before:
  - If it has length less than `tol`, determine whether it contains a root or a possible root.
  - Otherwise, process the interval using the Lipschitz constant.

  Return the result to thread 0 by sending:
  - `(3, a, b)` if a root has been found.
  - `(4, a, b)` if a possible root has been found.
  - `(5, a, b)` if the interval is guaranteed not to contain any roots.
  - `(6, a, c)`,
    `(7, fa, f(c))`,
    `(8, c, b)`,
    `(9, f(c), fb)` otherwise, where `c = (a+b)/2`.

Run your program for $k \in \{1, 2, 4, 8\}$. Report the roots and possible roots found, the time, and the speed-up.

2. Present and discuss your results.

2a. (4) Discuss the speed-ups (slow-downs?) of this program, the Matlab program, and the two previous C programs that you wrote, using the expensive function provided by Tyler for Homework 3. (Use Tyler's `AllSolve` codes instead of yours if you wish.)

2b (4) Rerun the 4 codes using `f(x) = sin(5x)` on the interval $[0.1, 3.3]$ (radians) and $k \in \{1, 2, 4, 8\}$. Demonstrate that all programs compute the same answer. Again compare the timings. In this case, overhead will show up more clearly.

2c (2) Compare the 4 programs for ease of writing, speed, and load balancing.

Notes:

- Use Pacheco's message passing programs in directory `omp_msg` of http://www.cs.umd.edu/users/oleary/c662/material/PachecoCh5code.zip. You will need to modify them (starting with `queue_lk.h`) to make the contents of a message an integer and two doubles instead of a single integer.
- Remember to demonstrate your mastery of writing good code, writing good documentation (including clear documentation of any changes you make to other people's programs), and having your program clearly label its output.