Questions to ponder:

1) Consider a multithreaded program to solve assignment #4. To be correct, the solution must be scalable to run on n processors. My computer is an eight-core processor. What results would you expect to see if the following executions of the program happen in order?

```
./CS460_SimpleThreads 1000000 2 1
./CS460_SimpleThreads 1000000 2 2
./CS460_SimpleThreads 1000000 2 4
./CS460_SimpleThreads 1000000 2 8
./CS460_SimpleThreads 1000000 2 16
./CS460_SimpleThreads 1000000 2 32
```

2) Thread Models

boss/worker – a single thread, the *boss* assigns work to other threads, the *workers*. Typically, the boss handles all input and parcels out work to the other tasks

pipeline - a task is broken into a series of suboperations, each of which is handled in series, but concurrently, by a different thread.

peer - similar to the manager/worker model, but after the main thread creates other threads, it participates in the work.

Which model(s) is(are) appropriate for the current assignment?

3) I have an extremely large word search puzzle and a dictionary of words. I've written the code to find all words from the dictionary that exist in the puzzle. My solution is single threaded and I would like to make the solution multithreaded. I have chosen to use the pipeline model where one thread looks horizontally and if not found passes the word to another thread that looks vertically and then two more threads if possible that look diagonally R-L and L-R.

a) Will this model work? Explain

b) Are there difficiencies with the model?

c) Is there a better model? If so, what is it?

4) When a new process moves from the ready state to the running state, each register in the CPU must be set to initial values or the values the process had when it was interrupted. Explain why the PC (program counter) register is typically the last register loaded for the new process.

5) The method run is called from two or more separate threads simultaneously. Is synchronization necessary? If so, illustrate your answer by using assembly instructions; otherwise, explain why not.

```
int gValue = 0;

void run(int value)

{

gValue += value;
return value;
}

// gValue is not accessed
// anywhere else in the code
```