ECE329 HW #5

Part 1: Implement a class called `Runnable` with the following public interface:

```
Runnable();          // creates a thread
~Runnable();         // waits for thread to complete, and cleans up
void Start();        // starts running thread;
                     // (if it is already running, then do nothing)
void Stop();         // notify thread to voluntarily stop running,
                     // and wait until the thread has completed before returning
```

The class should have

- a private method called `BaseRun` that is called when the thread starts:
  `DWORD BaseRun();`
- a private static method called `StaticThreadProc` that calls `BaseRun`:
  `static DWORD WINAPI StaticThreadProc( void* param );`
- a protected pure virtual method called `MainRoutine` that is called by `BaseRun` whenever someone calls `Runnable::Start`:
  `virtual DWORD MainRoutine() = 0;`

Part 2: Create a dialog-based application that compiles under Visual C++ 6.0 that has two buttons: `Start` and `Stop`, along with an edit box. When the user presses `Start`, a counter should be displayed in the edit box showing consecutive numbers, beginning with zero. When the user presses `Stop`, the counter should cease to increment (by telling the counter thread to stop *voluntarily*). When the user presses `Start` again, the counter should begin to increment, again beginning with the number zero. When the user presses `Ok` or `Cancel`, the counter should stop (if it is running), and the program should exit. The user should be able to press any button at any time, including successive presses of the same button (`Start` or `Stop`).

Your code should derive a class from the base `Runnable` class, and your `MainRoutine` should periodically call `StopRequested` to see whether it needs to exit. The purpose of this assignment is to give you more practice with synchronization primitives (mutexes and semaphores), so you should not create a new thread every time the `Start` button is pressed nor resume a suspended a thread. Instead, your thread should wait on your own synchronization primitives to accomplish this behavior.

Separately, answer the following problems in Chapter 4 of the Tanenbaum textbook:
5, 7, 8, 9, 10, 11, 20, 21 (erratum: memory size should be 256 MB).