Assigment 2: Signal Recognition

- Definition:
  - Apply input signal (a vector) to a set of precomputed transform matrices
  - Examine result to determine which of a collection of transform matrices is closest to signal
  - Compute M_1V, M_2V, …, M_nV
  - Revised formulation (for class purposes): compute MV_1, MV_2, …, MV_n

```
ApplySignal (float * mat, float *signal, int M) {
   float result = 0.0; /* register */

for (i=0; i<M; i++) {
   for (j=0; j<M; j++) {
      result[i] += mat[i][j] *signal[j];
}
```

Requirements:
- Use global memory, registers and shared memory only (no constant memory)
- Explore different ways of laying out data
- Explore different numbers of blocks and threads
- Be careful that formulation is correct

Assignment 2: What You Will Implement

We provide the sequential code. Your goal is to write two CUDA versions of this code:

(1) one that uses global memory

(2) one that uses a combination of global memory and shared memory

You'll time the code, but will not be graded on the actual performance. Rather, your score will be based on whether you produce two working versions of code, and the analysis of tradeoffs.

For your two versions, you should try three different thread and block decomposition strategies:

(1) a small number of blocks and a large number of threads

(2) a large number of blocks and fewer threads

(3) some intermediate point, or different number of dimensions in the block/thread decomposition

Assignment 2: Analyzing the Results

You'll need to perform a series of experiments and report on results. For each measurement, you should compute the average execution time of five runs.

What insights can you gain from the performance measurements and differences in behavior.

EXTRA CREDIT: Can you come up with a better implementation of this code? You can use other memory structures, or simply vary how much work is performed within a thread. How much faster is it?