There are some difficulties in doing the iterations of the stencil calculation using CUDA. Unfortunately, early versions of CUDA did not include a command to synchronize blocks, only a command to synchronize threads within a block. This is fixed in CUDA 9, but the system you are using only has CUDA 8. Here are 2 approaches to the problem (they aren't the only ones):

1. Have a loop in the CPU which calls the kernel, where the kernel only does 1 iteration. Remember to include cudaDeviceSynchronize
2. Call a "parent" kernel with a single block, where the parent has a loop that calls a "child" kernel which does 1 iteration. Since the parent has only 1 block, synchronizing its threads synchronizes an iteration. This is illustrated below.

To make the timing comparable between these options, do not use the CUDA timing routines, but instead use the standard system timer, starting the timer after the initial matrix has been copied to the device, and stopping after the verification values have been returned.

You will receive full credit for either approach. You can get a little extra credit if you use approach 2 and do an analysis of the best number of threads and blocks to use in the calls to the parent and to the child.

Using the parent/child approach:

```
__global__ void child(parameters) {
   work
}

__global__ void parent(parameters) {
    work, including loop which contains
       child<<<childblocks,childthreads>>>(parameters);
     __syncthreads();
}

int main() {
   initialize
   parent<<<1,parentthreads>>>(parameters);
   cudaDeviceSynchronize();
   finish
}
```

You can Google "dynamic parallelism in Cuda" for more details.

You need a new way to compile. You can use the following:

```
nvcc -arch=sm_35 -o hw5 hw5.cu -rdc=true -lcudadevrt
```