Q2

We implemented a tiled implementation by using the shared_block_size to use the shared memory to reduce the running time.

```
// Number of blocks and threads in 3 dimension
int xGrid = N / X_TILE_SIZE;
int yGrid = N / Y_TILE_SIZE;
int zGrid = N / Z_TILE_SIZE;

dim3 gridSize(X_TILE_SIZE, Y_TILE_SIZE, Z_TILE_SIZE);
dim3 blockSize(xGrid, yGrid, zGrid);

int shared_block_size = (X_TILE_SIZE + (2 * RADIUS))
                      * (Y_TILE_SIZE + (2 * RADIUS))
                      * (Z_TILE_SIZE + (2 * RADIUS)) * sizeof(double);

printf("starting CUDA code... \n");
start = CLOCK();
for (l = 0; l < LOOPS; l++) {
    tiling_mul<<< gridSize, blockSize, shared_block_size >>>(d_a, d_b, N);
}
```

Here is the running time of 3 different n values between tiled and non-tiled implementation.

n	non-tiled	tiled
32	0.345010 ms	0.331341 ms
64	2.155618 ms	1.971853 ms
128	15.526313 ms	15.104615 ms

```
[zhang.yam@d1004 Question2]$ ./Q2
starting CUDA code...
a result 105.6
The total time = 0.345010 ms
[zhang.yam@d1004 Question2]$ ./Q22
starting CUDA code...
a result 105.6
The total time = 0.331341 ms
```

```
[zhang.yam@d1004 Question2]$ ./Q2
starting CUDA code...
a result 105.6
The total time = 2.155618 ms
[zhang.yam@d1004 Question2]$ ./Q22
starting CUDA code...
a result 105.6
```

```
[zhang.yam@d1004 Question2]$ ./Q2
starting CUDA code...
a result 105.6
The total time = 15.526313 ms
[zhang.yam@d1004 Question2]$ ./Q22
starting CUDA code...
a result 105.6
The total time = 15.104615 ms
```

We could tell the running time of tiled implementation is faster but not so obvious.

(2). We could also change to a faster GPU to accelerate the running time of the calculation.

REFERENCE:

1. https://github.com/drewtu2/eece5640/blob/master/hw6/src/stencil_shared.cu