Problem 1: The code below is part of the matrix multiplication program used as one of the classroom examples. Based on a casual classroom suggestion the increment on c_idx was changed from thread_count to 1. Without any other changes that would result in multiple threads computing the same element of c (instead of just one thread computing a particular element of c).

For the questions below use an array size of $2^n \times 2^n$ elements and a block size of $2^b$ where $b \leq n$. If necessary, design your code for a device with eight multiprocessors.

(a) Modify the code so that the +1 increment remains, but each element of c is computed just once. **Hint: The solution will require a change to the initial value of c_idx and the final value.**

(b) For your solution determine the number of cache lines touched by a block for accesses to a and b. Show separate numbers for a and b. The numbers should be in terms of the array and block size.

```c
__global__ void mm_iter() {
    // Compute a unique index (number) for this thread.
    // This will be used as an array index.
    //
    int tid = threadIdx.x + blockIdx.x * blockDim.x;
    int thread_count = blockDim.x * gridDim.x;
    int row_mask = row_stride - 1;

    for ( int c_idx = tid; c_idx < array_size; c_idx += 1 )
    {
        int col = c_idx & row_mask;
        int row = c_idx >> row_stride_lg;
        int a_idx_base = row << row_stride_lg;
        float c_value = 0;
        for ( int k=0; k<row_stride; k++ )
        {
            int a_idx = a_idx_base + k;
            int b_idx = ( k << row_stride_lg ) + col;
            c_value += a[a_idx] * b[b_idx];
        }
        c[c_idx] = c_value;
    }
}
```