LSU EE 7700-2 Homework 2 Due: 6 February 2013

Problem 1: Consider the vertex transformation program presented in class:

```
#define N 5
typedef Elt_Type double;
struct Vertex { Elt_Type a[4]; };
struct App {
  int num_threads;
  Elt_Type matrix[4][4];
  int array_size;
  Vertex *v_in, *v_out;
} app;

void thread_start(void *arg) {
  const int tid = (ptrdiff_t) arg;
  const int elt_per_thread = app.array_size / app.num_threads;
  const int start = elt_per_thread * tid;
  const int stop = start + elt_per_thread;

  for ( int h=start; h<stop; h++ )
    {
      Vertex p = app.v_in[h];   Vertex q;
      for ( int i=0; i<4; i++ )
        {
          q.a[i] = 0;
          for ( int j=0; j<4; j++ ) q.a[i] += app.matrix[i][j] * p.a[j];
        }
      app.v_out[h] = q;
    }
}
```

(*a*) The code above is written for 4-element vertices. Generalize the code for $n-$element vertices. Use the macro `N` defined on the first line (which is set to 5, but could be set to other values). *Hint: The solution is fairly simple, and does not involve adding any lines of code, just modifying what's already there.*

(*b*) Compute the number of floating-point operations per vertex when $n = 5$. A multiply-add should be counted as one floating-point operation (the multiply and the add).

(*c*) Compute the amount of data transferred per vertex when $n = 5$. Consider both single- and double-precision numbers. (The code above is written for double-precision FP numbers, which are 8 bytes. Changing the `typedef` type to `float` will convert the code to use single-precision FP numbers, which are 4 bytes.)

(*d*) Repeat the problem for $n-$element vertices and an $n \times n$ matrix. Assume that $n \ll$ the number of vertices.

Problem 2: The NVIDIA GTX 690, a GPU meant for home use, has the following specifications:

- Memory bandwidth: 384 GB/s.
- Single-precision computation rate of 2.8 TFLOPS (2.8×10^{12} FP operations per second).
- Double-precision rate of 117 GFLOPS (yes, less than a 10th the single-precision rate).

(For the sake of simplicity in this problem a multiply/add instruction (FMADD) is counted as one floating-point operation.)

Consider the code from the previous problem.

(*a*) Based on these numbers determine the maximum number of $5-$element single-precision vertices per second that the GTX 690 can process.

(*b*) Repeat the problem above for double-precision elements.

(*c*) At what size vertex (value of n) will both the floating point computation rate and data transfer rate both equally limit the computation rate for the single-precision vertex program?

(*d*) Consider the results above. If you need the calculations performed in double precision, why shouldn't you complain to NVIDIA about the fact that the double-precision performance is less than $\frac{1}{10}$ that of single precision for smaller values of n? *Grading note: The original question was for $n = 5$, for which there was no reason to complain. See solution.*