1 Thread-Level Parallelism

As powerful as data level parallelization is, it can be quite inflexible, as not all applications have data that can be vectorized. Multithreading, or running a single piece of software on multiple hardware threads, is much more powerful and versatile. OpenMP provides an easy interface for using multithreading within C programs. Some examples of OpenMP directives:

The `parallel` directive indicates that each thread should run a copy of the code within the block. If a for loop is put within the block, every thread will run every iteration of the for loop.

```c
#pragma omp parallel {
    ...
}
```

The `parallel for` directive will split up iterations of a for loop over various threads. Every thread will run different iterations of the for loop. The following two code snippets are equivalent.

```c
#pragma omp parallel for
for (int i = 0; i < n; i++) {
    ...
}
```

There are two functions you can call that may be useful to you:

- `int omp_get_thread_num()` will return the number of the thread executing the code
- `int omp_get_num_threads()` will return the number of total hardware threads executing the code

For each question below, state and justify whether the program is sometimes incorrect, always incorrect, slower than serial, faster than serial, or none of the above. Assume the default number of threads is greater than 1. Assume no thread will complete before another thread starts executing. Assume `arr` is an `int[]` of length `n`.

(a) // Set element i of arr to i

```c
#pragma omp parallel
{
    for (int i = 0; i < n; i++)
        arr[i] = i;
}
```
(b) // Set arr to be an array of Fibonacci numbers.
   arr[0] = 0;
   arr[1] = 1;
   
   #pragma omp parallel for
   for (int i = 2; i < n; i++)
       arr[i] = arr[i-1] + arr[i - 2];

(c) // Set all elements in arr to 0;
   int i;
   #pragma omp parallel for
   for (i = 0; i < n; i++)
       arr[i] = 0;

1.2 What potential issue can arise from this code?

   // Decrements element i of arr. n is a multiple of omp_get_num_threads()
   #pragma omp parallel
   {
       int threadCount = omp_get_num_threads();
       int myThread = omp_get_thread_num();
       for (int i = 0; i < n; i++) {
           if (i % threadCount == myThread) arr[i] *= arr[i];
       }
   }

1.3 // Assume n holds the length of arr
   double fast_product(double *arr, int n) {
       double product = 1;
       #pragma omp parallel for
       for (i = 0; i < n; i++) {
           product *= arr[n];
       }
       return product;
   }

(a) What is wrong with this code?

(b) Fix the code using #pragma omp critical

(c) Fix the code using #pragma omp reduction(operation: var).
2 Logic Gates

2.1 Label the following logic gates:

2.2 Convert the following to boolean expressions:
   (a) NAND
   (b) XOR
   (c) XNOR

2.3 Create an AND gate using only NAND gates.

2.4 How many different two-input logic gates can there be? How many n-input logic gates?