Programming for Performance (ECE459): Pre-midterm Study Questions for Final

I'll choose one of these four questions to put on the final.

Answer the questions in your answer book. You may consult any printed material (books, notes, etc).

Question A: Restrict Qualifier

Write down a C function with `restrict`-qualified pointer arguments. Make sure that a compiler could profitably use the `restrict` qualifiers. Also write down a call to your C function which does not satisfy the conditions of the `restrict` qualifier, and discuss the possible results of that call.

Question B: Thread Pools

I may ask one of the following two questions:

- Show me an example of an incorrect execution that might occur if you wrote `if (queue.isEmpty())` instead of `while (queue.isEmpty())` while implementing a thread pool (see Lecture 10 notes).
- Incorporate some control logic into the skeleton thread pool implementation in Lecture 10. (I alluded to this in the notes, where I wrote that Java thread pools may supply a minimum and maximum number of threads.) I'm looking for pseudocode that makes sure that the right number of threads are active.

Question C: Reductions

We saw this example of a reduction in the notes.

```
1 double sum (double *array, int length)
2 {
3   double total = 0;
4
5   for (int i = 0; i < length; i++)
6     total += array[i];
7   return total;
8 }
```

I mentioned that the Solaris compiler could detect that it was a reduction and parallelize it. Write down, in reasonably-detailed pseudocode, the corresponding parallelized code, as well as the assumptions that you're making about the + operator in your parallelization.

Question D: Race Detection

Consider the following code (or code like it, on the exam; possibly a bit more complicated):

```
1  #include <pthread.h>
2  #include <stdio.h>
3
4  void f(int * a) {
5    *a = 4;
6  }
7
8  int * g() {
9    return malloc(sizeof(int));
10 }
11
12 void * h(void * p) {
13   int * q = (int *)p;
14   *q = 3;
15   printf("%d\n", *q);
16 }
17
18 int main() {
19   int * x = malloc(sizeof(int));
20   int * y = g();
21   pthread_t t;
22   *x = 2; *y = 2;
23
24   pthread_create(&t, NULL, h, x);
25   f(x);
26   f(y);
27   printf("%d %d\n", *x, *y);
28 }
```

Identify the race conditions in the code. Explain two different outputs that it could produce.