CSCI-UA.0480-003

Parallel Computing

Final Exam

Spring 2016 - May 16th (90 minutes)

NAME: **NetID:**

- If you have to make assumptions to continue solving a problem, state your assumptions clearly.
- You answer on the question sheet. You can use extra white papers if you want.

1. [1] We know that a block cannot be assigned to an SM until it gets all the resources it needs beforehand. What is the advantage of doing so?

2. We have seen that if-else may lead to branch divergence in a warp due to lockstep execution of instructions. Now, suppose there is a kernel that has an *if without else*.

a. [2] Can this also lead to **performance loss** in some cases, relative to non-branch divergence? Justify your answer. No need to write code, just explain.

b. [2] Can this also lead to **NO performance loss** in some cases, relative to non-branch divergence? Justify your answer. No need to write code, just explain.

3. [2] Can we have a race condition among threads belonging to the same warp? Justify your answer.

4. [6] For each variable in the following code: identify the scope of the variable, justify your choice, and for each variable identify potential race condition, if any. You can assume that a, b, c, i, N, and j have been defined somewhere before the parallel block.

```
#pragma omp parallel for private(a,b)
for (i = 0; i < N; i++) {
        int x = 0;
        c--;
        for (j = i; j < N; j++)
                x += func(c, b[j]);
        a[i] = x;
}
```

Variable	**Private/ shared**	**Why?**	**race cond? (Y/N)**	**Why?**
a[]				
b[]				
c				
i				
j				
x				

5. [2]State two shortcomings of Amadahl’s law.
    -
    -

6. For the following piece of code (assume very large number of cores):

```
…
int globalvalue = 0;
int main() {
        int numprocs, rank;
        int i = 0;

        MPI_Init(NULL, NULL);
        MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);

        #pragma omp parallel for shared(result) reduction(+:globalvalue)
        for( i = 0; i < 2+rank ; i++)
        {
          globalvalue ++;
          …rest of loop body …
        }

        MPI_Finalize();
 }
```

We execute the above code with: mpirun -n 4 *progname*

a. [2]How many threads we will end up having in the *whole system*? Explain.

b. [1] Just before executing MPI_Finalize(), how many instances of *globalvalue* do we have in the system?

c. [2] Is there a potential race condition in globalvalue++ ? Justify