CS 351 Exam 4, Fall 2011

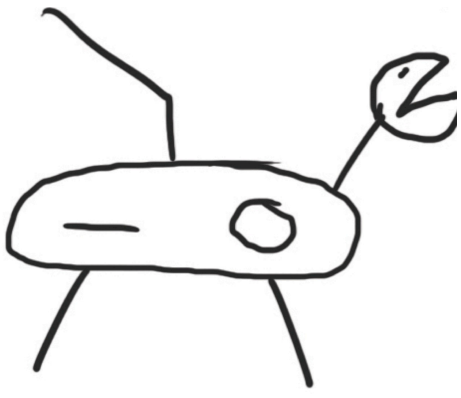
Your name: _____

Rules

- You may use one handwritten 8.5 x 11" cheat sheet (front and back). This is the only resource you may consult during this exam.
- Include explanations and comments in your answers in order to maximize your partial credit. However, you will be penalized for giving extraneous incorrect information.
- You may use the backs of these pages if you need more space, but make it clear where to find your answer to each question.
- Unless otherwise specified, you do not need to work out the arithmetic on math problems. Just do enough algebra to set up an answer of the form: Answer = [arithmetic expression] [units]

Grade:

	Your Score	**Max Score**
Problem 1		45
Problem 2		10
Problem 3		15
Problem 4		30
Total		100



Problem 1: Reliability, Disk Performance, and RAID (45 points)

Assume a dataset large enough to fill four 1-TB disks.

The average seek time of one of these disks is 5 ms, and the rotation speed is 15,000 rpm. The controller overhead is negligible. Each disk's transfer rate is 100 MB/s.

These disks handle two types of requests:

- Small requests – dominated by latency
- Large requests – 20 MB requests consisting of latency plus transfer time

---

a) [2 points] How many BYTES of data is this dataset? Assume we're using powers of 10, not powers of 2. You can use scientific notation.

b) [8 points] How long does it take to do a 20MB request if the data is stored consecutively on a single disk?

c) [8 points] In a certain RAID configuration, a disk fails once every 720 calendar days, and it takes one hour to reconstruct the data.

What is the MTTF of the array?

What is the MTBF?

What is the MTTR?

What is the availability?

d) [19 points] If we configure these disks in a RAID 5 array…

How many extra disks must we buy?

How long will it take to complete 20 small reads (assuming they are evenly distributed)? Explain.

How long will it take to complete 20 small writes (assuming they are evenly distributed)? Explain.

How long will it take to complete one large read?

How long will it take to complete one large write?

e) [8 points] RAID 4, unlike RAID 5, uses a dedicated parity disk. Which of the performance numbers above would be different if we used RAID 4? Show the new performance for these accesses.

Problem 2: I/O (10 points)

You have an important workload that scans through large chunks of sequential data, replacing an average of 1 byte for every MB that it reads.

---

a) [6 points] You are choosing between hard disks and flash for your workload. *Explain* which storage technology would work best in terms of…

- Performance?

- Cost (considering reliability and power consumption)?

b) [4 points] Would you use polling, interrupts, or DMA to handle the reads from disk? Why?

Problem 3: Parallel hardware (15 points)

Answer the following questions.

---

a. You are writing a program that launches multiple threads for interacting with different I/O devices. Your CPU can only support a few hardware threads, so your friend suggests porting your program to a GPU. Comment on this idea.

b. You are trying to implement a reduction tree on a GPU. Which of these two approaches is likely to perform better, and why?

Note: here *tid* is the ID of the thread within the block.

Kernel 1:

```
for(int pass=1; pass < blockDim.x; pass *= 2) {
     middle = blockDim.x / (2 * pass)
     if (tid < middle) {
          // The first half, 4th, etc. of the threads
         data[tid] += data[tid + middle]
     }
// wait for threads to catch up before going to next level
     __syncthreads();
}
```

Kernel 2:

```
for(int pass=1; pass < blockDim.x; pass *= 2) {
     if (tid % (2*pass) == 0) {
          // Every 2nd, 4th, etc. thread
          data[tid] += data[tid + pass];
     }
// wait for threads to catch up before going to next level
     __syncthreads();
}
```

Problem 4: Parallel Programming (30 points)

---

a. [5 points] Write a C++ for-loop which would be appropriate for static scheduling in OpenMP. Explain.

b. [5 points] How would this loop have to be different for dynamic scheduling to be the right approach?

c. [5 points] Explain the problem with the following parallelization. How might you get around the problem?

```
#pragma omp parallel for
for (int i=0; i < N-1; i++)
     A[i] += A[i+1]
```

d. [5 points] Draw a dependency graph with nodes for each statement.

```
A = f1(a, b) // Statement 1
B = f2(a, c) // Statement 2
C = f3(A, d) // Statement 3
D = f4(B, A) // Statement 4
A = f5() // Statement 5
```

[5 points[If each of the functions f_i is purely sequential, how many threads could this code be profitably split into?

[5 points] Are there any restrictions on reordering this code that are not captured in your dependency graph? Explain.