Name: _____

Directions: Work only on this sheet (on both sides, if needed); do not turn in any supplementary sheets of paper. There is actually plenty of room for your answers, as long as you organize yourself BEFORE starting writing. In order to get full credit, SHOW YOUR WORK.

1. (20) Consider our MPI example starting on p.10 of our notes, and suppose N = 25. How many messages of type PIPE_MSG will node 1 receive? What about node 2?

2. (20) Suppose we have a 64-node hypercube, and we sort using the Hyperquicksort algorithm shown on p.38 of our notes. Consider what occurs at node 13 on the second iteration of the loop. Suppose the array elements at that node are currently 12, 15, 29, 30, 35 and 48, while the elements at the root of this node's subcube are 14, 16 and 20. Which elements will node 13 send to another node, and what will be the node number of that node?

3. (15) In the JIAJIA example Primes.c, suppose we had forgotten the first lock/unlock pair within DoWork(). For each of the following statements, write 'P' (possible) or 'I' (impossible):

- The number of primes printed out in main() will be too large.
- The number of primes printed out in main() will be too small.
- The program will hang, at one or more nodes.
- The running time of the program will increase.
- The running time of the program will decrease.

4. (15) Consider an Ω-network consisting of 32 PEs. Through which switches will a read request for PE 4 to PE 21 pass?

5. (15) Suppose Intel were to add a fetch-and-add operation to its instruction set and cache coherency protocol. On p.27 of our notes there are tables for what happens when our CPU does a read or a write. Show what the corresponding table would be if our CPU does a fetch-and-add.

6. (15) In the JIAJIA example, say N = 10^6 and have 10 nodes. What will be the final value of NextI?

Solutions:

1. 8, 6

2. Here d = 6, i = 5. The 5-cube is all nodes of ID form 0. The root's media is 16. The two 4-cubes consist of all nodes of ID form 00 and 01, respectively. Node 13, i.e. node 001101, has 4-cube partner 011101, i.e. node 29. Node 13 then snes 29, 30, 35 and 48 to node 29.

3. As mentioned in class, the error may result in two (or more) nodes getting the same value of NextI. The fact that those nodes would do duplicate work wouldn't harm the accuracy of the program's results, but harm would come because it would mean that some values of NextI would be skipped. Then some numbers which should be crossed out, wouldn't be. That means the reported number of primes would be too large, though this wouldn't cause the program to hang anywhere. Without the locks, the program would speed up.

4. First, it will be convenient to note that S_{ij} has inputs $I_{i,2j}$ (left) and $I_{i,2j+1}$ (right), and outputs $O_{i,2j}$ (left) and $O_{i,2j+1}$ (right).

Node 4 launches its packet via I_{04}, which our comment above shows is connected to S_{02}.

Since the first bit of our destination 10101 is a 1, we turn right, i.e. leave S_{02}'s right output, O_{05}. Using the formula given in the notes, we see that O_{05} is connected to I_{1m}, where

$$m = (2 \cdot 5 + \lfloor 2 \cdot 5/32 \rfloor) mod 32 = 10$$

So, our packet enters $I_{1,10}$, which is in S_{15}.

We next turn left, due to a 0 bit in the destination ID, exiting via $O_{1,10}$ and then entering $I_{2,20}$. The latter is in $S_{2,10}$.

And so on, i.e. our packet goes through switches S_{02}, S_{15}, $S_{2,10}$, etc.

5. Since a fetch-and-add will consist of a read followed by an immediate write, we can just combine the existing read and write tables. So, the last column in the read table becomes M, M, E, M, E.

6. Each node will increment NextI once after the latter reaches 1000, so the final value will be 1010.