CSc 422/522 — Homework 4

due Tuesday, April 24

The first three problems are worth 10 points each. The last problem is worth 20 points. As usual, graduate students are to solve all problems (50 points), and undergraduates are to solve a combination that adds to 40 points.

1. MPD book, Exercise 7.6. You solution should be fair.

2. MPD book, Exercise 7.14.

3. MPD book, Exercise 7.17.

4. Write a distributed parallel program to solve one of the problems below. You may write your program in MPD, in C using the MPI library, or in Java using sockets. Develop your program on Lectura. If you choose the grid or primes problems, run your program on Par with 1, 2, 3, and 4 worker processes to see what kind of speedup you get. You do not have to beat on your program to get the best possible speedup (although you are welcome to do so :-). It is sufficient to write what you think is a good program.

Turn in (1) a commented listing of your program, (2) output from a representative set of runs, and (3) a brief explanation of your results. Also use `turnin` to submit a copy of your program. The assignment name is `hw4.prob4`. Just turn in the source file(s). Be sure your program reads its command-line arguments in the order specified below for each problem.

Problem 1: Distributed Grid Computation. Modify your parallel red/black program from the last project so that it uses message passing rather than shared variables. Try to do sends early and receives late so as to maximize the potential overlap between computation and communication. The command-line arguments are the same as for the shared program: `gridSize`, `numIters`, and `numWorkers`. The output should also be the same as before.

Problem 2: Primes using a Distributed Bag of Tasks. Write a parallel program to compute primes using the manager/workers paradigm described in Section 9.1 of the text. The two command-line arguments are `numWorkers` and `L`, the largest number to check. Your program should thus employ `numWorkers+1` processes. Each worker should use the standard *sequential* algorithm for checking whether an odd number is prime. The manager process should implement the bag and gather results from the workers. At the end of the program, print the total number of primes you found (don't forget `2`!), the last 10 primes, and the execution time of the program.

Problem 3: Rock/Scissors/Paper. Write a distributed program to simulate a three-person rock/scissors/paper game. Each player randomly chooses one of rock, scissors, or paper. Then the players compare their choices to see who "won." Rock smashes scissors, scissors cut paper, and paper covers rock. Award a player 2 points if it beats both the others; award two players 1 point each if they both beat the third; otherwise award no points. Then the players play another game.

Use one process for each player. The players must interact directly with each other; do *not* use an additional coordinator process. The command-line argument should be `numGames`, the number of games to play. At the end print the total points won by each player.

Problem 4: Roller Coaster Problem. Write a distributed program to solve the roller coaster problem you considered earlier in Homework 3. However, use *two* car processes and use message passing for all communication. The car processes should go around the track one after the other. One of your challenges is to figure out how to get passengers loaded onto the correct car.

The command-line arguments should be `C`, the capacity of a car; `n`, the number of passengers; `numTrips`, the number of trips each *car* takes; and `rideTime`, the time it takes for the car to go around the track. Note that the third argument is now `numTrips`, not `numRides` as before. You may assume that `n` is greater than `2*C`.

The output of your program should again be a trace of significant events that illustrate the order in which things happen.