CSc 422/522 — Homework 1, Part B

due Tuesday, February 9

Each problem is worth 10 points. Graduate students are to solve all problems (40 points); undergraduates are to solve any two problems (20 points).

Recall that the work you turn in must be your own. Explain clearly and succinctly what you are doing; don't just give an answer.

1. Consider the following program:

```
int x = 1, y = 1, z = 1;

co  x = y + 1;
//  ⟨ y = x + 2; ⟩
//  z = x + y;
//  ⟨ await (x > 0)
        x = 0; y = 0; z = 0; ⟩
oc
```

In the first and third processes, the atomic actions are reading and writing the individual variables. The second and fourth processes are each a single atomic action.

(a) How many possible histories are there for this program? (Namely, the total number of different execution orders.)

(b) What are the possible final values of each variable?

(c) Suppose the condition in the await statement is changed to `(x > 1)`. Is the program guaranteed to terminate? Explain.

(d) Suppose the condition in the await statement is changed to `(x > 2)`. Is the program guaranteed to terminate? Explain.

2. Write an iterative parallel program in SR to solve the following problem. Create an `n` × `n` matrix `m`, where `n` is a command-line argument. Initialize each element of the matrix to zero or one with equal probability; use one of SR's `random` functions (page 297) for this. Then perform two computations: (a) find the total number of ones in the matrix, and (b) determine whether the matrix is symmetric. A matrix is symmetric if it is equal to its transpose, namely if `m[i,j] = m[j,i]` for all elements.

Your parallel program should use `PR` processes, where `PR` is also a command-line argument. Divide the matrix into strips and assign one strip to each process. Each process should initialize its own strip of the matrix. You may assume that `n` is a multiple of `PR`. Do not assume that assignments to shared variables are atomic; only individual references are atomic. [Hint: Use final code to combine information computed by the processes.]

At the end of the program, write out the execution time of the processes, the number of ones in the matrix, and whether or not the matrix is symmetric. You may also want to write the matrix to a file so that you can examine the output while debugging the program.

Turn in a listing of your program together with output from a few runs. Use large values of `n`, at least 1000. Use values of `PR` that are between 1 and 4.

3. Write a recursive parallel program in SR to solve the adaptive quadrature problem described in Section 1.5 of the text. Start with the program on page 113, but change it so that it creates *at most* `PR` processes, where `PR` is a command-line argument. In particular, when there are already at least `PR` processes, use sequential recursion instead of parallel recursion.

Integrate the function `sin(x)*(x**2)` over the interval `0.0` to `10.0`. You may assume for this program that incrementing a global variable is an atomic action.

At the end of the program, write out the execution time, the number of processes that were created, the number of sequential recursive calls that were made, and the computed area under the function.

Turn in a listing of your program together with the output for a few runs with different values of `PR`

4. The only real way to determine the performance of a program is to execute it. However, it would often be nice to be able to predict—at least approximately—how well a program might perform before taking the time to implement and test it. This is especially true of parallel programs, where you might want to decide whether to bother or which of several possible parallelizations to try.

Your task for this problem is to make a start at creating a simple performance model for parallel programs. What I am looking for is a formula or formulas with variables that stand for the key components of performance: computation time, process creation overhead, I/O time, synchronization overhead, process termination overhead, and anything else you think is important. The values of variables would be the actual times for different components in a specific language executing on a specific machine. You will also want to capture the number of processes and number of processors in your model.

Develop a model. Define the variables you include and explain the role of each. Develop one or a few simple formulas that capture relations between the variables, explain the formulas, and describe how one would use them to predict performance.

In developing your model, draw from the experience you have gained so far in the class. For example, run your programs on Par and get timing results, then see if you can predict those results with your model. Another thing you can do is develop simple test programs to measure specific overheads. Appendix E of the SR book gives actual numbers for some of SR's features and describes how they were measured.

This problem is obviously open-ended and hence grading will be somewhat subjective. Just do your best; you should learn a lot from the effort.

Optional. If you would really like to dive into the last problem and do a thorough job on it, I am willing to have it count for 20 points. However, you must check with me first and tell me exactly what you would do.