CSc 422/522 — Homework 1, Part B

due Tuesday, February 8

Problems 1 to 4 are worth five points each. Problems 5 and 6 are worth 10 points each. Graduate students are to solve all problems (40 points); undergraduates are to solve any combination of problems that adds up to 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. MPD book, Exercise 2.13.

2. MPD book, Exercise 2.18.

3. MPD book, Exercise 2.20, parts (a), (c), and (e).

4. MPD book, Exercise 2.33.

5. Write a recursive parallel program in SR to compute an approximation to π as described in Exercise 1.6 of the text. Your program should have two command-line arguments: `EPSILON` and `DEPTH`.

Start with the algorithm shown on pages 18-19 and implement it in SR. Your procedure should keep making parallel recursive calls until the sum of the two smaller areas is within `EPSILON` of the sum of the two larger areas. (See `/home/cs522/SamplePrograms/quad.sr` for the source for the iterative quadrature program handed out in class.) If you get a run-time error message indicating that you have run out of memory, try reducing the stack size of each process. (The default is 40000 bytes; change it by using the `-S` option of `srl`, the SR linker.)

Next modify your program so that it uses a recursion depth of at most `DEPTH`. In particular, quit recursing if either you have an approximate area within `EPSILON` *or* you have already made `DEPTH` parallel recursive calls. Using `DEPTH` will enable you to control the number of processes that are created.

At the end of the program, write out the total number of procedure calls that were made and the computed area under the function. Turn in a listing of your second program together with the output from a few runs with different values of `EPSILON` and `DEPTH`. Briefly explain your results and what you learned from them.

6. Write an SR program that takes four, *sorted* input files and merges them into a single, *sorted* output file. The names of the input files should be command-line arguments. The program should write its result to the standard output file.

Use three processes in your program. One should merge the first two input files; one should merge the other two input files; and one should merge the results produced by the first two processes and write the final result. The processes are very similar to each other, and each is similar to the `shuffle` program you wrote in Part A of Homework 1.

Turn in a listing of your program together with a brief explanation of the tests you ran and the results you observed.