CS493-02 Cluster Computing
Final
8:30 am - 11:00 am, Thursday December 11th, 2003

Name: .....

Part I is closed book. Do not refer to any materials for this part. Return Part I to get Part II.

Total /50

Part I

Do not refer to any materials for this part

Qu. 1 Answer each of the following briefly:

(a) Suggest one reason that the multiprocessor speedup factor could be greater than *p* with a processor multiprocessor system. (There are several possible reasons - two in notes.)

2

(b) What is a message tag?

2

(c) Describe the actions of a scatter routine.

2

(d) Why is the Mandelbrot calculation an embarrassingly parallel computation?

2

(e) What is meant by the phrase “unsafe” when applied to an MPI program.

2

(f) How can the number of synchronization points be reduced in some synchronous computations (such as in the heat distribution problem)?

2

(g) Explain why the following program segments will not work for computing the sum of the elements of an array A:

```
FORALL i := 1 TO N DO
þþþsum := sum + A[i];
```

2

Re-write the code so that it will work given N = 10 and two processes.

2

(h) Describe Radix sort and suggest how it might be parallelized.

4

CS493-02 Cluster Computing
Final
8:30 am - 11:00 am, Thursday December 11th, 2003

Part II

Name: .....

You may refer to any materials for this part (but not others in the class). Use additional paper for your answer. Make sure your name is on every sheet.

Qu. 2 In some sorting algorithms, two fully sorted listed are combined ("merged") into a one sorted list and then separated into two sorted lists, where all the numbers in one list are less than all the numbers in the other list. This could be done with one process sending a number to the second process, starting at the largest number of its sorted list, and one process sending a number to the first process, starting at the smallest number of its list. Both processes place the number received in the correct place in their sorted lists. For example, suppose the two processes have the lists:

Process 1	Process 2
10	7
25	14
60	16
65	47
87	92

Process 1 send the number 87 to process 2, and process 2 send the number 7 to process 1 to get:

Process 1	Process 2
7	14
10	16
25	47
60	87
65	92

This is repeated with process 1 sending 65, and process 2 sending 14, to get:

Process 1	Process 2
7	16
10	47
14	65
25	87
60	92

Process 1 then send 60, and process 2 sends 16, to obtain:

Process 1	Process 2
7	47
10	60
14	65
16	87
25	92

The algorithm will terminate when the number sent by process 1 is less that the number sent by process 2.

Write a complete MPI program to perform these actions. Provide comments in your code to help the grader! If I do not understand the code, I will assume it is incorrect. 20