Exam Parallel Programming 20 January 2006
Department of Computer Science, Faculty of Sciences

1. What are the diameter, bisection width, and number of edges per node of a 3-dimensional mesh (lattice) topology that has 125 nodes in total?

2. Someone measures the performance of a communication-intensive parallel program on two different clusters; one cluster uses 1 GHz CPUs, the other cluster uses 3 GHz CPUs, but except for this difference the clusters are identical (same number of CPUs, amount of memory, local network, etc.). On which cluster does the parallel program obtain the best speedups? Explain your answer with simple calculations.

3. Explain why the usage of asynchronous message passing may lead to the problem of buffer overflows, while synchronous message passing does not have this problem.

4. Explain how a *select statement* can be used to implement a server process with the following property. The server contains an integer variable X (initialized to zero) and accepts two different types of messages, to increase resp. decrease the value of X. For both messages, the server returns a message to the sender with the new value of X. The value of X, however, should always be between zero and ten. Messages that try to decrease X below zero or increase it above ten should therefore not be serviced immediately but should be delayed until the above condition can be satisfied. Implement this server process with a select statement, using syntax from either SR, Ada, or (clearly explained) pseudo-code.

5. A fundamental problem with parallel programming is mutual exclusion synchronization. If different processes try to modify the same shared variable at the same time, the result may become unpredictable. Different languages provide different solutions to this problem. Describe and compare the approaches and language primitives that are used in the following languages: Java, Orca, Linda, and High Performance Fortran.

6. Given are an array A of N integers and a function TEST:

```
function TEST (x: integer): boolean
```

   Write a parallel program that determines if A contains at least 5 elements for which TEST returns TRUE (i.e., TEST(A(i)=TRUE). The function TEST takes a fixed amount (1 second) of computation time, independent of its parameter. The program *must* terminate (e.g., all processes should invoke an 'exit' system call) when 5 elements have been found (by all processes together) for which TEST returns TRUE; the processes thus have to communicate with each other. You are asked to write three different versions of the parallel program:

   (a) One version using MPI message passing

   (b) One version using SR, in particular SR's implicit message receipt ('proc') construct

   (c) One version using Orca's shared data objects

Your code need not be syntactically correct MPI, SR, or Orca; however, explain clearly how you exploit the primitives that these languages provide.

7. Write a parallel program in C/Linda that computes all prime numbers between 1 and N. The generated prime numbers should be stored in Tuple Space. The program should use the Replicated Workers approach, so each process (worker) should repeatedly get the next candidate number and test if it is prime. To determine efficiently if a candidate number X is prime, the process should check if X can be divided by smaller *prime* numbers, which can be read from Tuple Space (as soon as they have been generated). The program need not be syntactically correct C/Linda, but do make a clear distinction between formal and actual parameters in Tuple Space operations.

8. Consider the following Fortran program

```
integer ind(100)          ! an array of 100 integers
real X(100), Y(100)       ! arrays of 100 floating point numbers
do i = 1,100              ! for-loop
  ind(i) = rand(1,100)    ! a random integer number between 1 and 100
  Y(i) = random()         ! random floating point number
  X(i) = 0.0
enddo

do j = 1, 1000
   do i = 2, 100
     X(i) = (Y(ind(i)) + X(i-1)) / 2
   enddo
enddo
```

Someone wants to parallelize the program using HPF (High Performance Fortran). Explain why HPF is unsuitable for this type of application. Why is it difficult to design efficient HPF data-distribution directives for the program? Also explain what communication primitives are generated by the HPF compiler for this program.

9. The parallel Barnes-Hut algorithm for hierarchical N-body problems tries to improve the *data locality* of the parallel program. Explain why this is important and how the algorithm manages to improve data locality.

10. The Transposition-Driven Search (TDS) algorithm communicates very much but nevertheless it obtains nearly perfect speedups. Explain why this large communication overhead does not prevent the algorithm from obtaining a very high performance.

Points

1	2	3	4	5	6a	6b	6c	7	8	9	10
7	7	7	7	10	7	7	7	10	7	7	7

Total: 90 (+ 10 = 100)