Exam Parallel Programming 21 January 2005
Department of Computer Science, Faculty of Sciences

1. What is Amdahl's law?

2. Given below is the pseudo-code for a parallel algorithm that tries to solve the All-pairs Shortest Paths (ASP) problem, i.e. it computes the shortest route between any two cities, where the lengths of the direct routes between the cities are given by a NxN matrix C. The send/receive primitives used are FIFO-ordered, so messages between two nodes always arrive in the order they were sent.

   Explain why this algorithm is incorrect and may sometimes compute incorrect results, depending on the timing behavior of the program.

```
int lb, ub;          /* lower/upper bound for this CPU */
int rowK[N], C[lb:ub, N];       /* pivot row ; matrix */

for (k = 1; k <= N; k++) {
   if (k >= lb && k <= ub) {          /* do I have it? */
      rowK = C[k,*];
      for (p = 1; p <= nproc; p++) /* broadcast row */
         if (p != myprocid) SEND(p, rowK);
   } else
      RECEIVE_FROM_ANY(&rowK);         /* receive row */
   for (i = lb; i <= ub; i++)       /* update my rows */
      for (j = 1; j <= N; j++)
         C[i,j] = MIN(C[i,j], C[i,k] + rowK[j]);
}
```

3. What is a processor array (vector machine)? Explain how it differs from a multiprocessor and why it is programmed differently.

4. Explain how an efficient multicast primitive can be implemented on Myrinet by changing the software (firmware) of the Myrinet network interface cards. Why is such a multicast primitive faster than a normal spanning-tree multicast primitive on top of point-to-point message passing over Myrinet?

5. Many iterative parallel programs have to decide at the end of each iteration whether the program should terminate or continue. Each process therefore has to check whether its part of the computation satisfies the given convergence criterion. Next, the processes have to communicate to check if *all* processes satisfy the criterion; if so, the program terminates, else it continues.

   In Orca, this behavior can be expressed using a 'voting' object with the following specification:

```
OBJECT SPECIFICATION VotingObject;
   OPERATION Vote(YesOrNo: boolean);    # vote whether to terminate
   OPERATION AwaitDecision(): boolean; # outcome of the voting
END;
```

Each process first calls the operation Vote (indicating with the Boolean parameter whether or not it wants to terminate) and then calls AwaitDecision. The latter operation gives the outcome of the voting process: it returns 'true' if all processes want to terminate, and 'false' otherwise. The operation blocks until the outcome of the votes is known.

~~(a) Give the implementation of the VotingObject type in Orca; make sure that the~~ operation AwaitDecision returns as soon as the outcome of the votes is clear (i.e., one process votes 'no', or all processes have voted 'yes'). The object needs to work correctly for only 1 iteration; the number of processes (P) is known and fixed.

(b) Show how the same behavior can be expressed using Linda's Tuple Space (i.e., implement the procedures Vote and AwaitDecision using Linda's Tuple Space operations). For the Tuple Space operations, indicate clearly which parameters are actuals and which are formals.

6. Consider the following (synthetic) HPF program

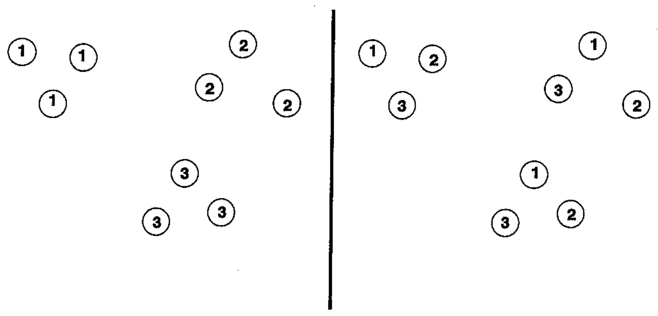
```
program hpfprogram
  real s, X(100), Y(100) ! s is scalar, X and Y are arrays
  !HPF$ PROCESSORS P(4)
  !HPF$ ALIGN X(:) WITH Y(:)
  !HPF$ ALIGN  s WITH Y(*)
  !HPF$ DISTRIBUTE Y(BLOCK) ONTO P

  X = X * 3.0        ! Multiply each X(i) by 3.0
  do i = 2,99
    Y(i) = X(i-1) + X(i+1)
  enddo
  s = SUM(X)   ! Add all X(i) values
end
```

(a) Explain how an HPF compiler will parallelize this program

(b) Explain which communication statements the HPF compiler will generate; assume that the compiler generates MPI calls and tries to make optimal use of the MPI primitives.

7. Consider an N-body system with 9 bodies, which will be simulated on 3 processors using the Barnes-Hut algorithm. The figure on the next page shows two different distributions of the 9 bodies over the 3 processors; the number in each body indicates the CPU to which the body is assigned.

Explain why the distribution may affect the performance of the parallel program. Which of the two distributions will give the best speedups? Why?



8. A programmer has implemented a parallel SOR (successive overrelaxation) program ~~based on the algorithm given below. The program obtains good speedups on a cluster~~ consisting of 64 Pentium-4 processors connected by a 1 Gbit/sec Myrinet network.

```
float G[lb-1:ub+1, 1:M], Gnew[lb-1:ub+1, 1:M];
for (step = 0; step < NSTEPS; step++)
   SEND(cpuid-1, G[lb]);       /* send 1st row left */
   SEND(cpuid+1, G[ub]);     /* send last row right */
   RECEIVE(cpuid-1, G[lb-1]);  /* receive from left */
   RECEIVE(cpuid+1, G[ub+1]); /* receive from right */
   for (i = lb; i <= ub; i++)     /* update my rows */
      for (j = 2; j < M; j++)
         Gnew[i,j] = (G[i,j]+G[i-1,j]+G[i+1,j]+G[i,j-1]+G[i,j+1])/5
   G = Gnew;
```

The programmer next tries to run this parallel program on a computational grid consisting of two identical clusters, one in Amsterdam and one in Paris, each with 32 Pentium-4 processors connected by Myrinet; the two clusters are connected by a wide-area network with a bandwidth of 10 Gbit/sec.

(a) Explain in detail why the speedup of the parallel SOR program will be worse on the two clusters than on one cluster.

(b) Try to think of an optimization that will improve the efficiency of the program on two clusters.

(c) The programmer does a similar experiment with a TDS (Transposition-Driven Scheduling) program but now notices hardly any performance difference between the execution times on 1 cluster and 2 clusters. Explain why this is the case.

Points

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

Total: 90 (+ 10 = 100)