Name: __________________________________

Directions: Work only on this sheet (on both sides, if needed); do not turn in any supplementary sheets of paper. There is actually plenty of room for your answers, as long as you organize yourself BEFORE starting writing. In order to get full credit, SHOW YOUR WORK.

1. (35) In the 8-processor butterfly barrier on p.166 of Wilkinson and Allen, suppose that P_2 goes down after completion of the operation $P_2 \leftrightarrow P_3$ in the first stage. Which of the processors, if any, will then wait for the barrier forever?

2. (35) The following code has as its goal edge detection. There are N rows and N columns in the image. Fill in the missing code. The variable ED will control what we might loosely describe as the "amount" of edge detection; the larger ED is, the sharper the contrast of the edges. For simplicity here, ignore the issue of how we deal with complex numbers.

```
for (R = 0; R < N; R++)
   for (C  = 0; C < N; C++)  {
      S = 0.0;
      for (J = 0; J < N; J++)
         for (K = 0; K < N; K++)
            S += x[J][K] *
              exp(-TwoPiI*(J*R+K*C)/N);
      X[R][C] = S;
   }
// missing code here
for (R = 0; R < N; R++)
   for (C  = 0; C < N; C++)  {
      S = 0.0;
      for (J = 0; J < N; J++)
         for (K = 0; K < N; K++)
            S += X[J][K] *
              exp(TwoPiI*(J*R+K*C)/N);
      x[R][C] = S;
   }
```

3. (30) Write a PerlDSM program which implements the pipelined bubble sort on p.275 of Wilkinson and Allen, in the paragraph which begins with "Bubble sort, as written..." (If you wish, you may write in JIAJIA instead of PerlDSM, but that would be more difficult.) Your program must be of SPMD form, with P_k handling case i = k in the outer **for** loop in the sequential code shown on that page. Your progam must be complete, except that you may omit the code which reads in the array to be sorted from disk. Keep your code short. **Make sure to write a draft of your code on scratch paper first, and then copy the clean version to your exam sheet.**

Solutions:

1. P_0, P_4, P_6

2.

```
for (R = 0; R < ED; R++)
   for (C = 0; C < ED; C++)
      X[R][C] = 0.0;
```

3. Outline (from newsgroup):

> The pure pipelined version (not odd-even transposition) works like this: In the sequential code, have processor k do the case i = k in the outer for loop. That processor then waits until processor k-1 has its value of j in the inner for loop reach 2; then processor k can start j = 0. Also, from then on, processor must make sure that its value of j stays at least 2 behind processor k-1's value of j.

In the grading, the key points considered were (a) whether the student set up j as a shared array (one element for each processing node), and (b) whether the student set up code for each node to wait for the previous one. The code for (b) would look something like

```
while ($J[$MyNode-1] < $J[$MyNode] + 2)  {
   ;
}
```