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. (15) Look at the discussion of *shearsort* in pages 277ff of Wilkinson and Allen. The text suggests performing a transpose operation, using a single call to an *all-to-all* routine. Assume that we have n processors and n data points, as in the discussion in the text. Data point (i,j), named x, will be at processor k*i+j, where $k = \sqrt{n}$. Give the MPI code to do the *all-to-all* operation, followed by a call to printf() to print out the value received. **(Don't forget the printf().)**

2. (15) Give the complete C code implementation of the pseudode

```
for all possible moves of square I do {
   Q = malloc(sizeof(struct BoardPos));
   fill in *Q according to this move;
```

in the handout on the 8-Squares Puzzle problem.

3. (15) Look at the cyclic-striped partitioning on p.317 of Wilkinson and Allen. Suppose our matrix is 16x16 and that we are using 4 processors. Consider what happens when P_1 is performing the last of its four calls to the broadcast function. Note that it is actually a multicast, going only to a subset of the four processors. State which processor(s) P_1 will be sending to in this fourth call

4. This problem concerns the pthreads examples, Workpile.c and Quicksort.c.

a. (10) The code in Workpile.c is meant to be generally applicable to task-farm problems, not just Quicksort.c. State which argument in which function declared in Workpile.c ties that general machinery to a specific problem such as *quicksort*.

b. (10) Show which line in Workpile.c checks the termination condition for the workpile operation.

5. Suppose we wish to do noise reduction on a certain image. (To simplify things, assume we are going to do this on just one row of pixels.)

a. (10) Look at the equations for X_k and x_k at the bottom of p.349 of Wilkinson and Allen. State how one the two equations would be altered.

b. (10) Of the master-slave and pipelined parallelizations discussed on pp.353-354 (applied in this case to the inverse transform), state why one of the two methods would be more appropriate in our setting here.

6. (15) Consider the pipelined parallelization of *bubble sort*, described in Figure 9.9 of Wilkinson and Allen. Suppose we are implementing this in MulSim, sorting an n-item global data array 'a' using n processors. In the code on p.275, P_i will handle case i in the outer loop. There will be a global array my_j, with my_j[i] stating which j in the inner loop P_i is currently working on. Write the MulSim code for this algorithm, omitting declarations, header-file includes and so on, but showing full detail of the algorithm itself. (Note: You should not need to use any lock variables.)

Solutions:

1.

```
MPI_Alltoall(&x,1,MPI_Int,y,1,MPI_Int,MPI_COMM_WORLD);
printf("%d\n",y[k*j+i]);
```

2.

```
Q = (struct BoardPos *) malloc(sizeof(struct BoardPos));
memcpy(Q,P,sizeof(struct BoardPos));
RowI = Q->Row[I];  ColI = Q->Col[I];
RowBlank = Q->Row[8];  ColBlank = Q->Col[8];
if (abs(RowI-RowBlank) == 1)
   Swap(&Q->Row[I],&Q->Row[8]);
else if (abs(ColI-ColBlank) == 1)
   Swap(&Q->Col[I],&Q->Col[8]);
```

4.a. Argument worker_proc in work_init().

4.b.

```
while(wp->n_pile != 0 || wp->n_working != 0)
```

5.a. We want to remove the high-frequency components, i.e. the "noisy" ones. So, in the sum for x_k, we sum only from j=0 to j=M for some M < N-1.

5.b. In **this setting**, the master-slave approach would be bad, since it would compute some values of X_k that would never be used. (Must get part (a) correct to receive credit for part (b).)

6.

```
me = CPU_NUM;
for (j = 0; j < me; j++)  {
   // must wait until the "upstream" node has past this j
   while (my_j[me-1] <= me+1) ;
   k = j+1;
```

```
   if (a[j] > a[k])  {
      temp = a[j];
      a[j] = a[k];
      a[k] = temp;
   }
   // signal "downstream" node
   my_j[me] = j;
}
```