Name: ____________________________

Directions:

Save your work frequently. **Your OMSI app must fill your laptop screen at ALL times.** No other electronic devices allowed. If you forgot your laptop or did not go through the OMSI dry run and have problems with the app, you must submit your work on the paper quiz sheet.

1. Name the three basic groups of wires in a basic system bus.

2. On p.21, it is mentioned that MPI accommmodates heterogeneous systems, e.g. in which some of the nodes are little-endian and some are big-endian. In class we discussed another type of heterogeneity accommmodated by MPI. What was it?

3. In the MPI example, pp.18ff, how many pairs of sockets are set up? How many are actually used? EXPLAIN.

4. Here is the code you used to test OpenMP on your machine:

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]) {
   int nthreads, tid;
   #pragma omp parallel private(nthreads, tid)
   {
     tid = omp_get_thread_num();
     printf("Hello World from thread = %d\n", tid);
     if (tid == 0) {
       nthreads = omp_get_num_threads();
       printf("Number of threads = %d\n", nthreads);
     }
   }
}
```

Use an OpenMP pragma to replace the "if," with the understanding that we no longer care which threads prints out the number of threads; we just need it so that that printing is done only once.

For full credit, your submission must be compilable and runnable. Try it! That is a major point of using OMSI.

5. Consider the **pthreads** example, pp.8ff. Suppose we did not want to have global variables, and wish to change the code accordingly. We start small, changing only **nextbase**. State what changes would need to be made.

Solutions:

1. data, address, control

2. In MPI send/receive operations, the sender and receiver may have different word sizes, say 64-bit and 32-bit. An **MPI_int** in one size will be converted to the other.

3. There is a manager and 3 workers, so C(4,2) = 6 socket pairs will be set up if we have a separate manager and 3 workers. But only 2 will be used, node 0 to node 1 and node 1 to node 2.

4.

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]) {
   int nthreads, tid;
   #pragma omp parallel private(nthreads, tid)
   {
     tid = omp_get_thread_num();
     printf("Hello World from thread = %d\n", tid);
     #pragma omp single
     {
       nthreads = omp_get_num_threads();
       printf("Number of threads = %d\n", nthreads);
     }
   }
}
```

5. Remove **nextbase** from line 24, and add it to line 64. In line 80, replace **i** by a **struct** containing **i** and ***nextbase**. In line 39, replace **int tn** by a pointer to the **struct**, and make similar changes within the body of **worker()**.