CPS 5401 Fall 2015
Shirley Moore, Instructor
Final Exam Practice Problems

1. Explain what is wrong with the OpenMP code below and modify the code to be correct. Assume that N and A have already been initialized.

```
max_val = A[0];
#pragma omp parallel for
for (i=1; i<N; i++)
{
     if (A[i] > max_val)
    {
          max_val = A[i];
    }
}
printf"\nmax_val = %f", max_val);
```

2. Suppose each core of an 8-core processor runs at 2GHz and that the memory bandwidth for the processor as measured by the multithreaded version of the STREAM benchmark is the following:

Number of cores	Measured memory bandwidth (GB/s)
1	10.2
2	20.3
3	31.0
4	42.1
5	50.5
6	61.0
7	54.7
8	51.2

   a. Assume that each processing core has a fused multiply-add instruction, that it can execute two such instructions in one cycle, and that the L1 data cache size for each core is 32KB. Assume memory operations can be completely overlapped with communication. Sketch the roofline model for the processor and label the machine balance point.
   b. Assume the processor characteristics given above. Consider the two-loop formulation of a 4K x 4K matrix times a 4K vector, where each matrix element is one word equal to 4 bytes. Assuming optimal cache placement, what is the peak performance obtainable for this computation on this processor, using up to but not necessarily all of the cores?

   c. What operational intensity (i.e., FLOPS/word) would be needed in order for a code to effectively utilize all eight cores of the processor described above? Please explain.

3. A correct OpenMP program will produce the same result as the equivalent serial program (i.e., if the OpenMP directives are ignored). Is the following OpenMP code correct? Why or why not?

```
int* arr = new int[10];
for(int i = 0; i < 10; i++)
    arr[i] = i;
#pragma omp parallel for
for (int i = 1; i < 10; i++)
    arr[i] = arr[i - 1];
for(int i = 0; i < 10; i++)
    printf("\narr[%d] = %d", i, arr[i]);
```

4. Is the MPI code below safe from deadlock? Why or why not? If not, fix it.

```
destrank = (myrank + 1) % numtasks;
srcrank = (myrank – 1 + numtasks) % numtasks;
MPI_Isend(sendbuf, 10, MPI_INT, destrank, 1, MPI_COMM_WORLD,
          &request);
MPI_Wait(&request, &status)
MPI_Recv(recvbuf, 10, MPI_INT, srcrank, 1, MPI_COMM_WORLD, &status);
```

5. A protein matching code takes four days to execute on a current machine. It spends 20% of the execution time doing integer instructions and 35% of the time doing I/O. For simplification, assume that I/O cannot be overlapped with computation. How would the performance improve with
   a. a compiler optimization that reduces the number of integers instructions by 25% (assuming an integer instructions always takes the same amount of time)?
   b. a hardware optimization that reduces the time for each I/O operation from 6us to 5us?
   c. applying both of the above optimizations?