Introduction to MPI - Worksheet 3 – Chapter 3
(See WebCT-HPC Course)

1. A *communicator* is a handle representing a group of processors that can communicate with each other. What is the MPI communicator that we are using?

2. There can be only one communicator in MPI. True/False.

3. Within each communicator, processors are numbered consecutively starting at ____. This number is known as the _____ of the processor.

4. If a processor belongs to more than one communicator, it's rank will still be the same number in each communicator. True/False.

5. "Size" means what for the communicator?

6. What 2 MPI routines return the rank and size?

7. The last MPI routine called in a program, the routine that terminates MPI is what?

8. **.** How would you modify "Hello World " so that only even-numbered processors print the greeting message? Write the code here:

9. To experience compiling and running a Fortran program, compile and run this version of the answer to #8 above. Run this program on all 16 processors of the SV1. Demonstrate the output – put this on your web portfolio.

   f90 -ffree hello.f NOTE: on the CRAY, use ! for the beginning of a comment line

   mpirun -np 16 a.out

10. Consider the following MPI pseudo-code, which sends a piece of data from processor 1 to processor 2:

```
MPI_INIT()
MPI_COMM_RANK(MPI_COMM_WORLD, myrank, ierr)
if (myrank = 1)
  MPI_SEND(some data to processor 2 in MPI_COMM_WORLD)
else {
  MPI_RECV(data from processor 1 in MPI_COMM_WORLD)
  print "Message received!"
}
MPI_FINALIZE()
```

   where MPI_SEND and MPI_RECV are blocking send and receive routines. Thus, for example, a process encountering the MPI_RECV statement will block while waiting for a message from processor 1.

   If this code is run on a single processor, what do you expect to happen?

   A. The code will print "Message received!" and then terminate.
   B. The code will terminate normally with no output.
   C. The code will hang with no output.
   D. An error condition will result.

11. Each process executes a copy of the entire code. in an MPI program. True/False

12. Point-to-point communication in MPI is "two-sided", meaning what?

13. MPI uses what three pieces of information in the “message body”?

14. What is the difference between blocking and nonblocking send/receive?

15. What is the difference between a synchronous and a buffered send?

16. Briefly describe a “broadcast” operation.

17. Briefly describe a “scatter” operation.

18. Briefly describe a “reduction” operation.

19.1. Point-to-point communication (check your answers online)
2. Collective communication 3. Communication mode 4. Blocking send
5. Synchronous send 6. Broadcast 7. Scatter 8. Gather

a. A send routine that does not return until it is is complete
b. Communication involving one or more groups of processes
c. A send routine that is not complete until receipt of the message at its destination has been acknowledged
d. An operation in which one process sends the same data to several others
e. Communication involving a single pair of processes
f. An operation in which one process distributes different elements of a local array to several others
g. An operation in which one process collects data from several others and assembles them in a local array
h. Specification of the method of operation and completion criteria for a communication routine

20. Which of the following is true for all send routines?
A. It is always safe to overwrite the sent variable(s) on the sending processor after the send returns.
B.Completion implies that the message has been received at its

destination.

C. It is always safe to overwrite the sent variable(s) on the sending processor after the send is complete.

D. All of the above.

E. None of the above.

21. Is a blocking send necessarily also synchronous? Yes/No
Briefly explain why.

22.Consider the following fragment of MPI pseudo-code:

```
...
x = fun(y)
MPI_SOME_SEND(the value of x to some other processor)
x = fun(z)
...
```

where MPI_SOME_SEND is a generic send routine. In this case, it would be best to use

A. A blocking send

B. A nonblocking send

21. Explain your reasoning for #20 above.