The following questions ask you to analyze some code fragments and to write some code fragments. When you analyze some code, your analysis should be written in complete sentences organized into paragraphs. Do not write sentence fragments and do not write the most terse answer that you can think of (even if it is essentially correct). You are being graded on your ability to communicate, not just on your ability to arrive at correct solutions.

When you write code fragments, you do not need to write compilable code. Just make sure that your code is not in any way ambiguous.

Write all of your answers neatly on separate pieces of paper. If you want, you can write your answers using a text editor, Word, or even LaTeX. If your answers are all contained in an electronic document (e.g., text file, Word document, pdf file, etc.), then you can submit your exam to me by Blackboard. If you write up your solutions long hand, then turn in your exam in my mailbox in the Mathematics Department office or in class.

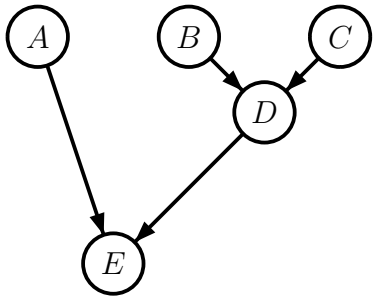
Each person should work on this exam by them self. If you have any questions about the exam, feel free to send me an e-mail or call me on the phone.

This exam should be turned in on Thursday, December 13.

1. Suppose that we have five C functions

```
void functionA(void);
void functionB(void);
void functionC(void);
void functionD(void);
void functionE(void);
```

that together solve some problem (strictly by using side effects, since there are no parameters or return values). Suppose these function depend on each other according to the following graph.



Each edge of the graph denotes a dependency between two of these functions. For example, the edge from node B to node D means that `functionB` must be called, and must return, before `functionD` can be called.

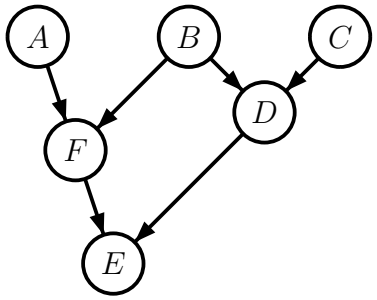
Write a C program that uses MPI to execute the above five functions in a way that is maximally parallel, but adheres to the above dependency graph. (Don't worry about how the five functions do their side effects when they are on different processors. Just assume that the side effects happen.)

Solution:

2. Suppose that we have six C functions

```
void functionA(void);
void functionB(void);
void functionC(void);
void functionD(void);
void functionE(void);
void functionF(void);
```

that together solve some problem. Suppose these function depend on each other according to the following dependency graph.



Write a C program that uses MPI to execute the above six functions in a way that is maximally parallel, but adheres to the above dependency graph.

Solution:

3. Here is a sketch of a possible way that an implementation of MPI might implement the `MPI_barrier()` function. This implementation of `MPI_barrier()` is in terms of the more primitive MPI functions `MPI_Send()` and `MPI_Recv()`.

```
int MPI_Barrier(MPI_Comm comm)
{  int p, rank, i, mesg = 0, tag = 1;
   MPI_Status status;
   MPI_Comm_size(comm, &p);
   MPI_Comm_rank(comm, &rank);
   if (rank == 0)
   {
      for(i = 1; i < p; i++)
         MPI_Recv(&mesg, 1, MPI_INT, i, tag, comm, &status);
      for(i = 1; i < p; i++)
         MPI_Send(&mesg, 1, MPI_INT, i, tag, comm);
   }
   else
   {
      MPI_Send(&mesg, 1, MPI_INT, 0, tag, comm);
      MPI_Recv(&mesg, 1, MPI_INT, 0, tag, comm, &status);
   }

   return 0;
}
```

(a) Why is this a poor way to implement the barrier function? (Hint: This implementation is too serialized.)

(b) Give a sketch of a way to implement at least a slightly better version of the barrier function (but still in terms of the more primitive MPI functions `MPI_Send()` and `MPI_Recv()`). Explain how your version is better. (Hint: Find a way to "parallelize" the implementation of the barrier.)

Solution: