Name: _____

Directions: MAKE SURE TO COPY YOUR ANSWERS TO A SEPARATE SHEET FOR SENDING ME AN ELECTRONIC COPY LATER.

1. (100) Below is MPI code for transforming an adjacency matrix, as in Section 4.13. Fill in the blanks.

```
// transforming an adjacency matrix,
// MPI version

#include <mpi.h>

int nwrkrs,  // number of workers
    // number of vertices, assumed
    // divisible by # of workers
    nv,
    me,  // my node number
    *adj,  // the adjacency matrix
    *xmat,  // transformed matrix
    finaloutrownum;
// rows in xmat when done

void init(int argc,char **argv)
{  int i,j,tmp;
   nv = atoi(argv[1]);
   MPI_Init(&argc,&argv);
   MPI_Comm_size(MPI_COMM_WORLD,&tmp);
   nwrkrs = tmp - 1;
   MPI_Comm_rank(MPI_COMM_WORLD,&me);
   adj = malloc(_____);  // blank (a)
   // as test, fill adj with random 0s,1s
   for (i = 0; i < nv; i++)
      for (j = 0; j < nv; j++)
         adj[i*nv+j] = rand() % 2;
   if (me == 0 && nv < 10)
      for (i = 0; i < nv; i++) {
         for (j = 0; j < nv; j++)
            printf("%d ",adj[nv*i+j]);
         printf("\n");
      }
}

void mgr()
{  int i,
       chunksize = nv / nwrkrs,
       outrownum = 0,
       nrecv;
   MPI_Status status;
   xmat = malloc(_____);  // blank (b)
   int maxrecv = _____  ;  // blank (c)
   for (i = 1; i <= nwrkrs; i++)  {
      MPI_Recv(_____  ,  // blank (d)
         maxrecv,MPI_INT,i,
         MPI_ANY_TAG,MPI_COMM_WORLD,&status);
      MPI_____(  // blank (e)
         &status,MPI_INT,&nrecv);
      outrownum += _____  // blank (f)
   }
   finaloutrownum = outrownum;
}

void wrkr()
{  int chunksize = nv / nwrkrs,
       outrownum = 0,
       mystartrow, myendrow,i,j;
   xmat = malloc(chunksize*nv*2*sizeof(int));
   mystartrow = _____ // blank (g)
   myendrow = _____ // blank (h)
   for (i = mystartrow; i <= myendrow; i++)
      for (j = 0; j < nv; j++) {
         if (adj[nv*i+j] == 1) {
            xmat[2*outrownum] = i;
            xmat[2*outrownum+1] = j;
            __________ // blank (i)
(one full C stmt)
         }
      }
   MPI_Send(xmat,_____,  // blank (j)
      MPI_INT,0,0,MPI_COMM_WORLD);
}

int main(int argc,char **argv)
{  int i,j;
   init(argc,argv);
   if (me == 0) mgr();
   else wrkr();
   if (me == 0 && nv < 10)
      for (i = 0; i < finaloutrownum; i++) {
         for (j = 0; j < 2; j++)
            printf("%d ",xmat[2*i+j]);
         printf("\n");
      }
   MPI_Finalize();
}
```

Solutions:

1.

```
// transforming an adjacency matrix,
// MPI version

#include <mpi.h>

int nwrkrs,  // number of workers
    // number of verts.; assumed div. by # of workers
    nv,
    me,  // my node number
    *adj,  // the adjacency matrix
    *xmat,  // transformed matrix
    finaloutrownum;  // rows in xmat when done

void init(int argc,char **argv)
{  int i,j,tmp;
   nv = atoi(argv[1]);
   MPI_Init(&argc,&argv);
   MPI_Comm_size(MPI_COMM_WORLD,&tmp);
   nwrkrs = tmp - 1;
   MPI_Comm_rank(MPI_COMM_WORLD,&me);
   adj = malloc(nv*nv*sizeof(int));
   // as test, fill adj with random 0s,1s
   for (i = 0; i < nv; i++)
      for (j = 0; j < nv; j++)
         adj[i*nv+j] = rand() % 2;
   if (me == 0 && nv < 10)
      for (i = 0; i < nv; i++) {
         for (j = 0; j < nv; j++)
            printf("%d ",adj[nv*i+j]);
         printf("\n");
      }
}

void mgr()
{  int i,
       chunksize = nv / nwrkrs,
       outrownum = 0,
       nrecv;
   MPI_Status status;
   xmat = malloc(nv*nv*2*sizeof(int));
   int maxrecv = chunksize * nv * 2;
   for (i = 1; i <= nwrkrs; i++)  {
      MPI_Recv(xmat+outrownum*2,maxrecv,MPI_INT,i,
         MPI_ANY_TAG,MPI_COMM_WORLD,&status);
      MPI_Get_count(&status,MPI_INT,&nrecv);
      outrownum += nrecv / 2;
   }
   finaloutrownum = outrownum;
}

void wrkr()
{  int chunksize = nv / nwrkrs,
       outrownum = 0,
       mystartrow, myendrow,i,j;
   xmat = malloc(chunksize*nv*2*sizeof(int));
   mystartrow = (me-1) * chunksize;
   myendrow = mystartrow + chunksize - 1;
   for (i = mystartrow; i <= myendrow; i++)
      for (j = 0; j < nv; j++) {
         if (adj[nv*i+j] == 1) {
            xmat[2*outrownum] = i;
            xmat[2*outrownum+1] = j;
            outrownum++;
         }
      }
   MPI_Send(xmat,outrownum*2,MPI_INT,0,0,MPI_COMM_WORLD);
}

int main(int argc,char **argv)
```

```
{   int i,j;
    init(argc,argv);
    if (me == 0) mgr();
    else wrkr();
    if (me == 0 && nv < 10)
       for (i = 0; i < finaloutrownum; i++) {
          for (j = 0; j < 2; j++)
             printf("%d ",xmat[2*i+j]);
          printf("\n");
       }
    MPI_Finalize();
}
```