Name: ____________________________

Directions: **Work only on this sheet** (on both sides, if needed). MAKE SURE TO COPY YOUR ANSWERS TO A SEPARATE SHEET FOR SENDING ME AN ELECTRONIC COPY LATER.

IMPORTANT NOTE: If you believe that nothing needs to be placed into a blank, simply give **Nothing** as your answer.

1. (50) Consider the mutual outlinks example, beginning on p.93. Below is a different version of **dowork()**. One of the differences is that it uses dynamic loop scheduling. Another difference is that it doesn't use **atomic** or **critical**. Fill in the blanks.

```
float dowork()
{
   #pragma omp parallel
   {  int i;
      #pragma omp for BLANKa
      for (BLANKb) {
         tot += procpairs(i);
      }
   }
   BLANKc
   BLANKd
   int divisor = n * (n-1) / 2;
   return ((float) tot)/divisor;
}
```

2. (50) Below is an MPI program that removes 0s from an array. The strategy is that first the manager node breaks the original array into equal-sized chunks, sending one for each worker. Each worker removes the 0s and sends back the nonzero elements. The manager collects these into an array **no0s**. Fill in the blanks below:

```
#include <mpi.h>
#include <stdlib.h>
#define MAX_N 100000
#define MAX_NPROCS 100
#define DATA_MSG 0
#define NEWDATA_MSG 1

int nnodes,  // number of MPI processes
    n,  // size of original array
    me,  // my MPI ID
    has0s[MAX_N],  // original data
    no0s[MAX_N],  // 0-free data
    nno0s;  // number of non-0 elements
int debug;

// not shown
init(int argc, char **argv)

void managernode()
{
   MPI_Status status;
   int i;
   int lenchunk;
   // assumed divides evenly
   lenchunk = n / nnodes;
   for (i = 1; i < nnodes; i++) {
      BLANKa
   }
   int k = 0;
   for (i = 1; i < nnodes; i++) {
      BLANKb
      BLANKc
      BLANKd
   }
   nno0s = k;
}

// not shown
void remov0s(int *oldx, int n, int *newx, int *nnewx)

void workernode()
{
   int lenchunk;
   MPI_Status status;
   BLANKe
   BLANKf
   remov0s(has0s,lenchunk,no0s,&nno0s);
   BLANKg
}

// not shown
int main(int argc,char **argv)
```

Solutions:

1.

```
float dowork()
{
   #pragma omp parallel
   {  int i;
      #pragma omp for reduction(+:tot) schedule(dynamic)
      for (i = 0; i < n-1; i++) {
         tot += procpairs(i);
      }
   }
   int divisor = n * (n-1) / 2;
   return ((float) tot)/divisor;
}
```

2.

```
#include <mpi.h>
#include <stdlib.h>

#define MAX_N 100000
#define MAX_NPROCS 100
#define DATA_MSG 0
#define NEWDATA_MSG 1

int nnodes,  // number of MPI processes
    n,  // size of original array
    me,  // my MPI ID
    has0s[MAX_N],  // original data
    no0s[MAX_N],  // 0-free data
    nno0s;  // number of non-0 elements

int debug;

init(int argc, char **argv)
{
   int i;
   MPI_Init(&argc,&argv);
   MPI_Comm_size(MPI_COMM_WORLD,&nnodes);
   MPI_Comm_rank(MPI_COMM_WORLD,&me);
   n = atoi(argv[1]);
   if (me == 0) {
      for (i = 0; i < n; i++)
         has0s[i] = rand() % 4;
   } else {
      debug = atoi(argv[2]);
      while (debug) ;
   }
}

void managernode()
{
   MPI_Status status;
   int i;
   int lenchunk;
   lenchunk = n / (nnodes-1);  // assumed divides evenly
   for (i = 1; i < nnodes; i++) {
      MPI_Send(has0s+(i-1)*lenchunk,lenchunk,
         MPI_INT,i,DATA_MSG,MPI_COMM_WORLD);
   }
   int k = 0;
   for (i = 1; i < nnodes; i++) {
      MPI_Recv(no0s+k,MAX_N,
         MPI_INT,i,NEWDATA_MSG,MPI_COMM_WORLD,&status);
      MPI_Get_count(&status,MPI_INT,&lenchunk);
      k += lenchunk;
   }
   nno0s = k;
}

void remov0s(int *oldx, int n, int *newx, int *nnewx)
```

```
{  int i,count = 0;
   for (i = 0; i < n; i++)
      if (oldx[i] != 0) newx[count++] = oldx[i];
   *nnewx = count;
}

void workernode()
{
   int lenchunk;
   MPI_Status status;
   MPI_Recv(has0s,MAX_N,
      MPI_INT,0,DATA_MSG,MPI_COMM_WORLD,&status);
   MPI_Get_count(&status,MPI_INT,&lenchunk);
   remov0s(has0s,lenchunk,no0s,&nno0s);
   MPI_Send(no0s,nno0s,
      MPI_INT,0,NEWDATA_MSG,MPI_COMM_WORLD);
}

int main(int argc,char **argv)
{
   int i;
   init(argc,argv);
   if (me == 0 && n < 25) {
      for (i = 0; i < n; i++) printf("%d ",has0s[i]);
      printf("\n");
   }
   if (me == 0) managernode();
   else workernode();
   if (me == 0 && n < 25) {
      for (i = 0; i < n; i++) printf("%d ",no0s[i]);
      printf("\n");
   }
   MPI_Finalize();
}
```