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 in your file. If you do not answer at all, put 00 in your file.

1. Here we will use MPI to find cumulative sums, as we did in Quiz 4 with CUDA. For instance, if the original array is (3,1,2,0,3,0,1,2), then it is changed to (3,4,6,6,9,9,10,12).

The original array starts out life at Node 0, which distributes chunks of it to the other nodes, though keeping one chunk for itself. A node will find cumulative sums for its chunk, and then adjust them based on the high values of the chunks that precede it. In the above example, for instance, say we have 4 nodes. The nodes will first produce (3,4), (2,2), (3,3) and (1,3). Since Node 0 found a cumulative sum of 4 in the end, we must add 4 to each element of (2,2), yielding (6,6). Etc.

Fill in the blanks.

```
1  // finds cumulative sums in the array x
2
3  #include <mpi.h>
4  #include <stdlib.h>
5
6  #define MAX_N 10000000
7  #define MAX_NODES 10
8
9  int nnodes,  // number of MPI processes
10     n,  // size of x
11     me,  // MPI rank of this node
12     // full data for node 0, part for the rest
13     x[MAX_N],
14     // cumulative sums for this node, and
15     // eventually for full array at Node 0
16     csums[MAX_N],
17     // max values at the various nodes
18     maxvals[MAX_NODES];
19
20 int debug;
21
22 init(int argc, char **argv)
23 {  // not shown, nothing special here
24 }
25
26 void cumulsums()
27 {
28    MPI_Status status;
29    int i,lenchunk,sum,node;
30    // assumed divides evenly
31    lenchunk = n / nnodes;
32    // note that node 0 will participate
33    // in the computation too
34    BLANKa(BLANKb, lenchunk, MPI_INT,
35       BLANKc, lenchunk, MPI_INT,
36       0,MPI_COMM_WORLD);
37    sum = 0;
38    for (i = 0; i < lenchunk; i++) {
39       csums[i] = sum + x[i];
40       BLANKd;
41    }
42    BLANKe(BLANKf,1,MPI_INT,BLANKg,1,MPI_INT,
43       0,MPI_COMM_WORLD);
44    BLANKh(BLANKi, nnodes, MPI_INT,0,MPI_COMM_WORLD);
45    if (me > 0) {
46       sum = 0;
47       for (node = 0; node < BLANKj; node++) {
48          sum += BLANKj[node];
49       }
50       for (i = 0; i < lenchunk; i++)
51          csums[i] += sum;
52    }
53    BLANKk(csums,lenchunk,MPI_INT,csums,lenchunk,MPI_INT,
54       0,MPI_COMM_WORLD);
55 }
56
57 int main(int argc,char **argv)
58 {  // not shown, nothing special here, other
59    // than a call to cumulsums()
60 }
```

Solutions:

```
1  // finds cumulative sums in the array x
2
3  #include <mpi.h>
4  #include <stdlib.h>
5
6  #define MAX_N 10000000
7  #define MAX_NODES 10
8
9  int nnodes,  // number of MPI processes
10     n,  // size of x
11     me,  // MPI rank of this node
12     // full data for node 0, part for the rest
13     x[MAX_N],
14     csums[MAX_N],  // cumulative sums for this node
15     maxvals[MAX_NODES];  // the max values at the various nodes
16
17 int debug;
18
19 init(int argc, char **argv)
20 {
21    int i;
22    MPI_Init(&argc,&argv);
23    MPI_Comm_size(MPI_COMM_WORLD,&nnodes);
24    MPI_Comm_rank(MPI_COMM_WORLD,&me);
25    n = atoi(argv[1]);
26    // test data
27    if (me == 0) {
28       for (i = 0; i < n; i++)
29          x[i] = rand() % 32;
30    }
31    debug = atoi(argv[2]);
32    while (debug) ;
33 }
34
35 void cumulsums()
36 {
37    MPI_Status status;
38    int i,lenchunk,sum,node;
39    lenchunk = n / nnodes;  // assumed to divide evenly
40    // note that node 0 will participate in the computation too
41    MPI_Scatter(x,lenchunk,MPI_INT,x,lenchunk,MPI_INT,
42       0,MPI_COMM_WORLD);
43    sum = 0;
44    for (i = 0; i < lenchunk; i++) {
45       csums[i] = sum + x[i];
46       sum += x[i];
47    }
48    MPI_Gather(&csums[lenchunk-1],1,MPI_INT,
49       maxvals,1,MPI_INT,0,MPI_COMM_WORLD);
50    MPI_Bcast(maxvals,nnodes,MPI_INT,0,MPI_COMM_WORLD);
51    if (me > 0) {
52       sum = 0;
53       for (node = 0; node < me; node++) {
54          sum += maxvals[node];
55       }
56       for (i = 0; i < lenchunk; i++)
57          csums[i] += sum;
58    }
59    MPI_Gather(csums,lenchunk,MPI_INT,csums,lenchunk,MPI_INT,
60       0,MPI_COMM_WORLD);
61 }
62
63 int main(int argc,char **argv)
64 {
65    int i;
66    init(argc,argv);
67    if (me == 0 && n < 25) {
68       for (i = 0; i < n; i++) printf("%d ",x[i]);
69       printf("\n");
70    }
```

```
71      cumulsums();
72      if (me == 0 && n < 25) {
73         for (i = 0; i < n; i++) printf("%d ",csums[i]);
74         printf("\n");
75      }
76      MPI_Finalize();
77   }
```