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. The CUDA code below converts an array to cumulative sums. 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 general plan is for each thread to operate on one chunk of the array. A thread 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 threads. The threads will first produce (3,4), (2,2), (3,3) and (1,3). Since thread 0 found a cumulative sum of 4 in the end, we must add 4 to each element of (2,2), yielding (6,6). Thread 1 had found a cumulative sum of 2 in the end, which together with the 4 found by thread 0 makes 6. Thus thread 2 must add 6 to each of its elements, i.e. add 6 to (3,3), yielding (9,9). The case of thread 3 is similar.

Fill in the blanks.

```
1  // CUDA example
2
3  // inputs an int array x of length n and
4  // computes cumulative sums of the
5  // elements, writing them back to x
6
7  // for this simple illustration, it is
8  // assumed that the code runs in just
9  // one block, and that the number of threads
10 // evenly divides n
11
12 #include <cuda.h>
13 #include <stdio.h>
14
15 __global__ void cumulker(int *dx, int n)
16 {
17    int me = BLANKa;
18    int csize  = n / BLANKb;
19    int start = BLANKc * csize;
20    int i,j,base;
21    for (i = 1; i < csize; i++) {
22       BLANKd;
23       dx[j] = dx[j-1] + dx[j];
24    }
25    BLANKe;
26    if (BLANKf) {
27       base = 0;
28       for (j = 0; j < me; j++)
29          BLANKg;
30    }
31    BLANKh;
32    if (me > 0) {
33       for (BLANKi)
34          dx[i] += base;
35    }
36 }
37
38 int cumul(int *x, int n, int bsize)
39 {
40    int *dx;  // device copy of x
41    BLANKi;
42    BLANKj;
43    dim3 dimGrid(1,1);
44    dim3 dimBlock(BLANKk,1,1);
45    cumulker<<<dimGrid,dimBlock>>>(dx,n);
46    BLANKl;
47    cudaMemcpy(x,dx,n*sizeof(int),cudaMemcpyDeviceToHost);
48    cudaFree(dx);
49 }
50
51 int main(int argc, char **argv)
52 {  int i;
53    int *x;  // host output matrix
54    int n = atoi(argv[1]);
55    int bsize = atoi(argv[2]);
56    x = (int *) malloc(n*sizeof(int));
57    for (i = 0; i < n; i++)
58          x[i] = rand() % 5;
59    if (n < 25) {
60       for (i = 0; i < n; i++) printf("%d ",x[i]);
61       printf("\n");
62    }
63    cumul(x,n,bsize);
64    if (n < 25) {
65       for (i = 0; i < n; i++) printf("%d ",x[i]);
66       printf("\n");
67    }
68 }
```

Solutions:

1.

```
1  // CUDA example
2
3  // inputs an int array x of length n and computes cumulative sums of the
4  // elements, writing them back to x
5
6  // for this simple illustration, it is assumed that the code runs in
7  // just one block, and that the number of threads evenly divides n
8
9  #include <cuda.h>
10 #include <stdio.h>
11
12 __global__ void cumulker(int *dx, int n)
13 {
14    int me = threadIdx.x;
15    int csize  = n / blockDim.x;
16    int start = me * csize;
17    int i,j,base;
18    for (i = 1; i < csize; i++) {
19       j = start + i;
20       dx[j] = dx[j-1] + dx[j];
21    }
22    __syncthreads();
23    if (me > 0) {
24       base = 0;
25       for (j = 0; j < me; j++)
26          base += dx[(j+1)*csize-1];
27    }
28    __syncthreads();
29    if (me > 0) {
30       for (i = start; i < start + csize; i++)
31          dx[i] += base;
32    }
33 }
34
35 int cumul(int *x, int n, int bsize)
36 {
37    int *dx;  // device copy of x
38    int chunksize = n / bsize;  // number of elements handled by each thread
39    cudaMalloc((void **)&dx,n*sizeof(int));
40    cudaMemcpy(dx,x,n*sizeof(int),cudaMemcpyHostToDevice);
41    dim3 dimGrid(1,1);
42    dim3 dimBlock(bsize,1,1);
43    cumulker<<<dimGrid,dimBlock>>>(dx,n);
44    cudaThreadSynchronize();
45    cudaMemcpy(x,dx,n*sizeof(int),cudaMemcpyDeviceToHost);
46    cudaFree(dx);
47 }
48
49 int main(int argc, char **argv)
50 {  int i;
51    int *x;  // host output matrix
52    int n = atoi(argv[1]);
53    int bsize = atoi(argv[2]);
54    x = (int *) malloc(n*sizeof(int));
55    for (i = 0; i < n; i++)
56          x[i] = rand() % 5;
57    if (n < 25) {
58       for (i = 0; i < n; i++) printf("%d ",x[i]);
59       printf("\n");
60    }
61    cumul(x,n,bsize);
62    if (n < 25) {
63       for (i = 0; i < n; i++) printf("%d ",x[i]);
64       printf("\n");
65    }
66 }
```