Write a C program that displays the following:

a. The number of CUDA devices available on your computer.

b. The number and names of CUDA devices having more than 256 multiprocessors.

a

```
int count;
cudaGetDeviceCount(&count);
printf("%d ", count);
```

b

```
~~int count;~~ Cuda Device Properties prop;
~~Cuda Get Device Count(&count);~~
int i; int num=0;
for (i=0; i < count; i++) {
    CudaGetDeviceProp(&prop, i);
    if(prop.multiprocessorCount > 256) {
        num += 1;
        printf("%s ", prop.name);
    }
}
printf("%d", num);
```

3

King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC453 – Parallel Processing – Midtterm 2 Exam – Fall 2021

Question 2

Let's consider 2 *N by N* square matrices of integers A and B. Let's consider that we would like to write a C program that runs in parallel and that computes and returns the sum of the 2 matrices:

C[i][j] = A[i][j] + B[i][j]

1. We would like to run this kernel within a grid composed of a single 2-D thread-block where every thread processes a single cell of the matrix.

   - Give the code of the following kernel.

     *__global__ void add(int *a, int *b, int *c, int N) {*

```
int rowIndex = ThreadIdx.y ;
int ColIndex = ThreadIdx.x ;
  C[rowIndex][ColIndex] = A[rowIndex][colIndex] +
  B[rowIndex][colIndex] ;
```

3

2. We would like to run this kernel within a grid composed of many 2-D thread-blocks where every thread processes a single cell of the matrix.

- Give the code of the kernel.

*__global__ void add(int *a, int *b, int *c, int N) {*

```
int rowIndex = ThreadIdx.y + blockIdx.y * blockDim.y;
int colIndex = ThreadIdx.x + blockIdx.x * blockDim.x;
C[rowIndex][colIndex] = A[rowIndex][colIndex] + ~~B[rowIndex][~~
                        B[rowIndex][colIndex];
```

3

}

3. We would like to run this kernel within a grid composed of many 2-D thread-blocks where every thread processes a sub-square matrix of size *W by W* .

- Give the code of the kernel.

*__global__ void add(int *a, int *b, int *c, int W, int N) {*

```
int rowIndex = (ThreadIdx.y + blockIdx.y * blockDim.y) * w;
int colIndex = (ThreadIdx.x + blockIdx.x * blockDim.x) * w;
int i; int j;
for (i = 0; i < w; i++) {
  for (j = 0; j < w; j++) {
C[rowIndex+i][colIndex+j] = A[rowIndex+i][colIndex+j]
  + B[rowIndex+i][colIndex + j]; } }
```

5

King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC453 – Parallel Processing – Midterm 2 Exam – Fall 2021

Question 3

Let's consider that we would like to sort ascendingly an array of integers of size N using the bitonic merge-sort algorithm.

1. Give the number of steps that are required to sort the elements of the array.

1 Steps $= \log_2(N)$

2. Give the number of stages that are required in every step *i*.

Stages $= i$

1 Same number as the step number

3. Give the size of bitonic sequences in every step.

1 2^{step}

4. Give the size of bitonic sequences in every stage of a step *i*.

1 $\frac{2^i}{2^{stage-1}}$

5. Give the condition that should satisfy a thread to participate in the processing of bitonic sequences of a stage *j* of a step *i*.

$$\text{ThreadIndex} \;\%\; \frac{2^i}{2^{j-1}} < \frac{\frac{2^i}{2^{j-1}}}{2}$$

1

6. Give the condition that should satisfy a thread that participates in the processing of sequences of a stage *j* of a step *i* to sort its corresponding bitonic-sequence ascendingly.

1 $$\frac{\text{ThreadIndex}}{2^i} \;\%\; \underline{2} == 0$$