King Saud University

Course Code:	CSC 453
Course Title:	Parallel Processing
Semester:	Spring 2024
Exercises Cover Sheet:	**Midterm 2 Exam**

Student Name:	[redacted]
Student ID:	
Student Section No.	

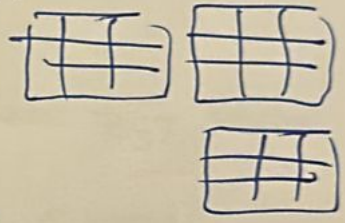
Question 1

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

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

1. We would like to run this kernel by a 2-D grid of thread-blocks each of which is 2-D array of threads 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 N, int W) {
int row = [threadIdx.y + (blockIdx.y * blockDim.y)] * W
int col = [threadIdx.x + (blockIdx.x * blockDim.x)] * W
for(int i=0; i<W; i++){
  for(int j=0; j<W; j++){
C[row+i][col+j] = A[row+i][col+j] + B[row+i][col+j]
}}
return 0; }
```

}

Question 2

1. Let's consider the following serial loop:

```
for (int i = 0 ; i < N; i ++)
            doSomething(data[i]);
```

   a. Give a parallel solution using N blocks with 1 thread each.

```
__global__ void X(N, data) {
index = blockIdx.x
if (index < N) { doSomething(data[index]); } }
```

2

   b. Give a parallel solution using 1 block with N threads.

```
__global__ void X(N, data) {
index = threadIdx.x
if (index < N) { doSomething(data[index]); } }
```

2

   c. Which solution from a and b performs well? Why?

(b) is better, bc in (b) we have 1 block with N threads $\Rightarrow \frac{N}{32}$ warps

and in (a) we have N block and 1 thread so $\Rightarrow$ N warps

So, with less warps means less cycles 2

2. Let's consider the following serial nested loops:

```
for (int i = 0 ; i < N; i ++)
    for (int j = 0 ; j < M; j ++)
        doSomething(i, j);
```

a. Give a parallel solution using a 1-D grid of 1-D thread blocks.

```
-global-void X(N, M){
int i = blockIdx.x
int j = threadIdx.x
if (i<N && j<M){
  doSomething(i, j);}}
```

2

b. Give a parallel solution using a 2-D grid of 2-D thread blocks.

```
-global-void X(N, M){
int R = threadIdx.y+blockIdx.y * blockDim.y
int C = threadIdx.x +blockIdx.x * blockDim.x
if(R<N && C<M){
    doSomething(R,C);}}
```

4

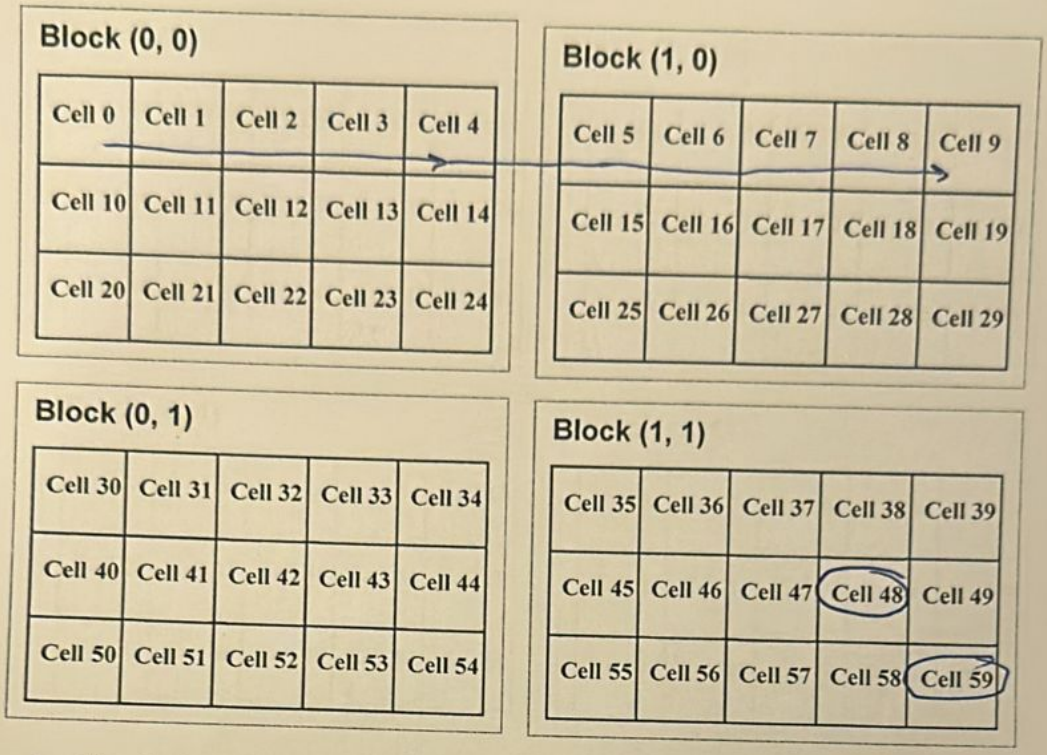
Question 3

Let's consider 2 arrays of integers A and B of size *N*. Let's consider that we would like to write a kernel using CUDA that computes in parallel the sum of the 2 arrays A and B:

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

```
__global__ void add(int *A, int *B, int *c,  int N) {
        int  cell_id = ..................;
        if (cell_id < N)
                C[cell_id] = A [cell_id] + B [cell_id] ;
}
```

1. We would like to run this kernel on grid composed of **multiple 2-D thread blocks**. Every thread evaluates a single cell as shown in the following figure:



- Give the formula that allows every thread to compute the cell_id of the cell he is going to process.

[blockIdx.y * gridDim.x * blockDim.x * blocDim.y]

+

[threadIdx.y * gridDim.x * blocDim.x]

+

[blockIdx.x * blockDim.x]

+

threadIdx.x

4

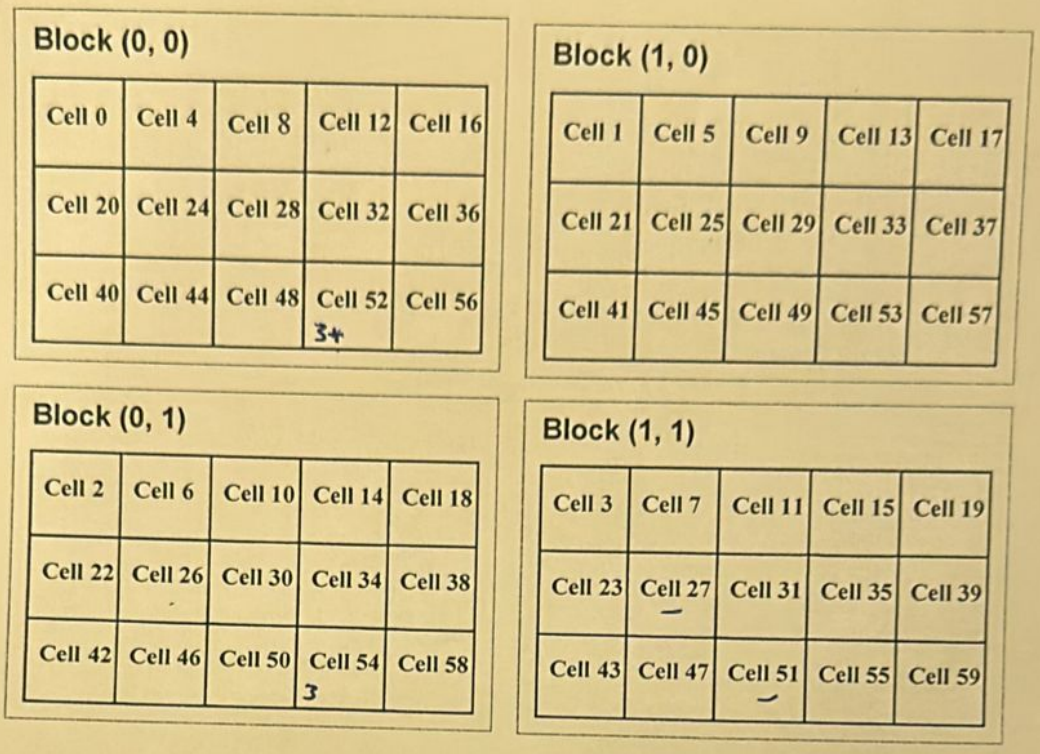
ex.

cell 59 = [1 x 2 x 5 x 3] + [2 x 2 x 5] + [1 x 5] + 4 = 59

30 + 20 + 5 + 4

50 + 9

2. We would like to run this kernel on **2-D grid of blocks** each of which is of **2-D array of threads**. Every thread evaluates a single cell as shown in the following figure:



a. Give the formula that allows every thread to compute the cell_id of the cell he is going to process.

threadIdx.x + [threadIdx.y * blockDim.x] * gridDim.x * gridDim.y
+ blockIdx.x + [blockIdx.y * gridDim.x]

ex.

Cell 52 = 3 + [2 x 5] x 2 x 2 + 0 + [0 x 2] = 52

Cell 49 = 2 + [2 x 5] x 2 x 2 + 1 + [0 x 2] = 49

Cell 54 = 3 + [2 x 5] x 2 x 2 + 0 + [1 x 2] = 54

Cell 59 = 4 + [2 x 5] x 2 x 2 + 1 + [1 x 2] = 59