King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC453 – Parallel Processing – Tutorial No 5bis – Quarter 3 2023

Question 1

Let's consider 2 arrays of integers A and B of size ***N***. Let's consider that we would like to write a C program that runs in parallel and that computes the sum of 2 arrays as following:

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

Let's consider the following kernel:

```
__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 **2-D grid of blocks** each of which is of **2-D matrix of threads**. Every thread evaluates a single cell as shown in the following figure:

Block (0, 0)

Cell 0	Cell 4	Cell 8	Cell 12	Cell 16
Cell 20	Cell 24	Cell 28	Cell 32	Cell 36
Cell 40	Cell 44	Cell 48	Cell 52	Cell 56

Block (1, 0)

Cell 1	Cell 5	Cell 9	Cell 13	Cell 17
Cell 21	Cell 25	Cell 29	Cell 33	Cell 37
Cell 41	Cell 45	Cell 49	Cell 53	Cell 57

Block (0, 1)

Cell 2	Cell 6	Cell 10	Cell 14	Cell 18
Cell 22	Cell 26	Cell 30	Cell 34	Cell 38
Cell 42	Cell 46	Cell 50	Cell 54	Cell 58

Block (1, 1)

Cell 3	Cell 7	Cell 11	Cell 15	Cell 19
Cell 23	Cell 27	Cell 31	Cell 35	Cell 39
Cell 43	Cell 47	Cell 51	Cell 55	Cell 59

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

Question 2

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

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

We would like to run this kernel on a grid composed of **1 block** where every thread evaluates ***W*** cells as shown in the following figure (where ***W*** = 4 as a sample):

Block (0, 0)

Cells 0-3	**Cells** 4-7	**Cells 8-11**	**Cells** 12-15	**Cells** 16-19
Cells 20-23	**Cells** 24-27	**Cells** 28-31	**Cells 32-35**	**Cells 36-39**
Cells 40-43	**Cells** 43-47	**Cells 48-51**	**Cells** 52-55	**Cells** 56-60

- Write the kernel

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

    int index = thredIdx.y * blockDim.x * w +
                threadIdx.x * w;
    for (int i = index; i < index + w; i++)
    C[i] = A[i] + B[i];
    }
```

King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC453 – Parallel Processing – Tutorial No 5bis – Quarter 3 2023

```
__global__ void kernal(int N, int *arr){
int index = blockIdx.x * blockDim.x + threadIdx.x
if (index < N) doSomthing(arr[index]);
}
```

Question 3

1. Let's consider an array of integers of size ***N,*** denoted Data. Let's consider the following sequential iteration:

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

```
__device__ void doSomthing(int a){...}
```

```
() *w
for(int i = index; i< index+w; i++)
```

   a. Write the corresponding parallel code using CUDA C.
   b. Update your kernel such that every threads operates W cells.

2. We would like to write a kernel that (1) receives an integer ***A***, two arrays of size ***N*** denoted ***X*** et ***Y*** respectively and (2) calculates and stores the ***A*X + Y*** in an array of size ***N*** denoted ***C***. We consider that the elements of the array ***C*** are calculated as follows:

$$C[i] = A * X[i] + Y[i]$$

So, we would like to write a kernel that receives 5 parameters:

- An integer denoted **A** (**input**),
- An array of integers denoted **X** (**input**),
- An array of integers denoted **Y** (**input**),
- An array of integers, denoted **C** (**output**),
- An integer denoted **N** which represent the size of arrays X, Y and C (**input**),

```
__global__ void kernal(int a, int *X, int *Y, int *C, int N)
int i = blockIdx.x*blockDIm.x + threadIdx.d;
if (i<N) C[i] = a * X[i] + Y[i];
```

   a. We would like to run this kernel within a grid of blocks (organized as 1-D array) each of which is a 1-D array of threads. We would like that <u>every thread calculates a single cell</u> of the result (**the array C**). Give the code of the kernel.

   b. We would like to run this kernel within a grid of blocks (organized as 1-D array) each of which is a 1-D array of threads. We would like that <u>every thread calculates the elements of a sub-array (**of the array C**) of size ***width***</u>. Give the code of the kernel.

```
() *w
for(int i = index; i< index+w; i++)
```