**King Saud University**
**College of Computer and Information Sciences**
**Department of Computer Science**
**CSC453 – Parallel Processing – Tutorial No 4 – Spring 2021**

# Question 1

1. Let's consider 2 integer Arrays A and B of dimension N. Let's consider that we would like to write a C program that runs in parallel and that computes the sum of the 2 arrays:

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

   a. Write the kernel (called ***kernel_1***) that will run on 1 Block of N threads.

```
__global__ void kernal_1(int *A, int *B, int *C, int N){
    int i = threadIdx.x;
    if(i < N) C[i] = A[i] + B[i];
}
```

   b. Write another kernel (called ***kernel_2***) that will run on N blocks with 1 thread each.

```
    int i = blockIdx.x;
```

   c. Write the main program that will call both kernels.

```
#define N 1024
int main(void){

    int *a, *b, *c;
    int *d_a, *d_b, *d_c;

    int size = N * sizeof(int);

    cudaMalloc((void**)&d_a, size);
    cudaMalloc((void**)&d_b, size);
    cudaMalloc((void**)&d_c, size);

    a = (int*)malloc(size); random_ints(a, N);
    b = (int*)malloc(size); random_ints(b, N);
    c = (int*)malloc(size);

    cudaMemcpy(d_a, a, size, cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, size, cudaMemcpyHostToDevice);

    kernal_1<<<1, N>>>(d_a, d_b, d_c);
    kernal_2<<<N, 1>>>(d_a, d_b, d_c);

    cudaMemcpy(c, d_c, size, cudaMemcpyDeviceToHost);

    free(a, b, c);
    cudaFree(d_a, d_b, d_c);

    return 0;
}
```

**King Saud University**
**College of Computer and Information Sciences**
**Department of Computer Science**
**CSC453 – Parallel Processing – Tutorial No 4 – Spring 2021**

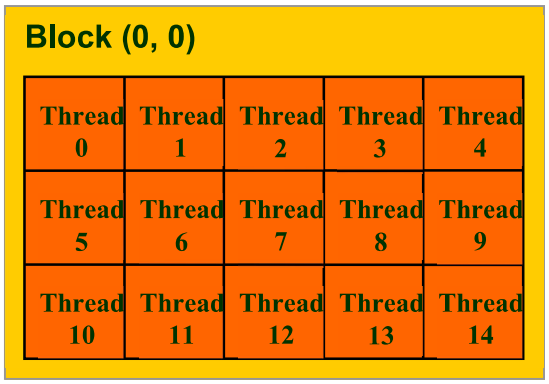
# Question 2

Let's consider 2 integer Arrays A and B of dimension N. Let's consider that we would like to write a C program that runs in parallel and that computes the sum of the 2 arrays:

C[index] = A[index] + B[index];

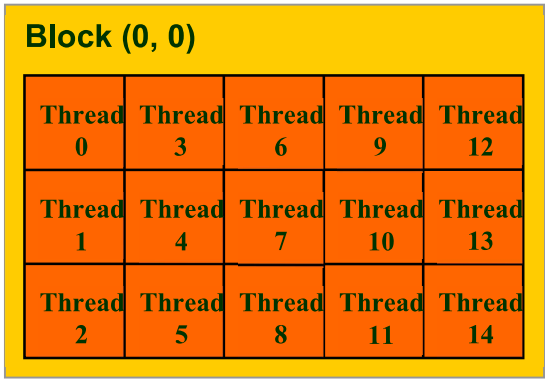
For every configuration of the grid of thread blocks described below, give the statement that computes the index for each each thread:

1. The grid is composed of 1 block and threads should have ids as in the following figure:




```
int index = threadIdx.y * blockDim.x + threadIdx.x;
```

2. The grid is composed of 1 block and threads should have ids as in the following figure:




```
int index = threadIdx.y +
            threadIdx.x * blockDim.y
```

2 + 3*3