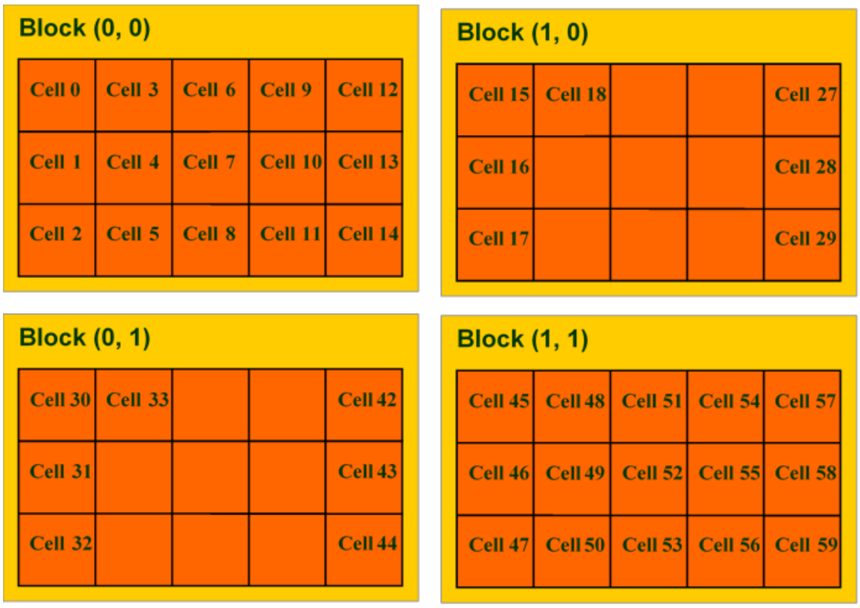
We would like to run a kernel on grid configured as M * N matrix of thread blocks. Every thread handles only one cell. Give the statement that calculates the **cell_id** for each thread as shown in the following figure:


Block (0, 0)
Cell 0 Cell 3 Cell 6 Cell 9 Cell 12
Cell 1 Cell 4 Cell 7 Cell 10 Cell 13
Cell 2 Cell 5 Cell 8 Cell 11 Cell 14
Block (1, 0)
Cell 15 Cell 18 Cell 27
Cell 16 Cell 28
Cell 17 Cell 29
Block (0, 1)
Cell 30 Cell 33 Cell 42
Cell 31 Cell 43
Cell 32 Cell 44
Block (1, 1)
Cell 45 Cell 48 Cell 51 Cell 54 Cell 57
Cell 46 Cell 49 Cell 52 Cell 55 Cell 58
Cell 47 Cell 50 Cell 53 Cell 56 Cell 59

Cell_id=(threadIdx.y+threadIdx.x*blockDim.y)+((blockIdx+blockIdx.x*gridDim.y)*(blockDim.x*blockDim.y));

Give the CUDA statements that allow to get the number of available GPU devises.

Int count;

cudaGetDeviceCount(&count);

printf(" the number of available GPU devises is %d ",count)

Give the statements that displays the name and the number of multi-processors of the current GPU device .

```
Int dev;

cudaDeviceProp prop;

cudaGetDevice(&dev);

cudaGetDeviceProperties(%prop,dev);

printf(“name of the current GPU device is:%s /n”,prop.name);

printf(“number of multi-processors is: %d/n
”,prop.MultiProcessorCount);
```

Let's consider a (N by N) Matrix of integers . We assume that the space memory on the GPU device for the Matrix is already allocated.

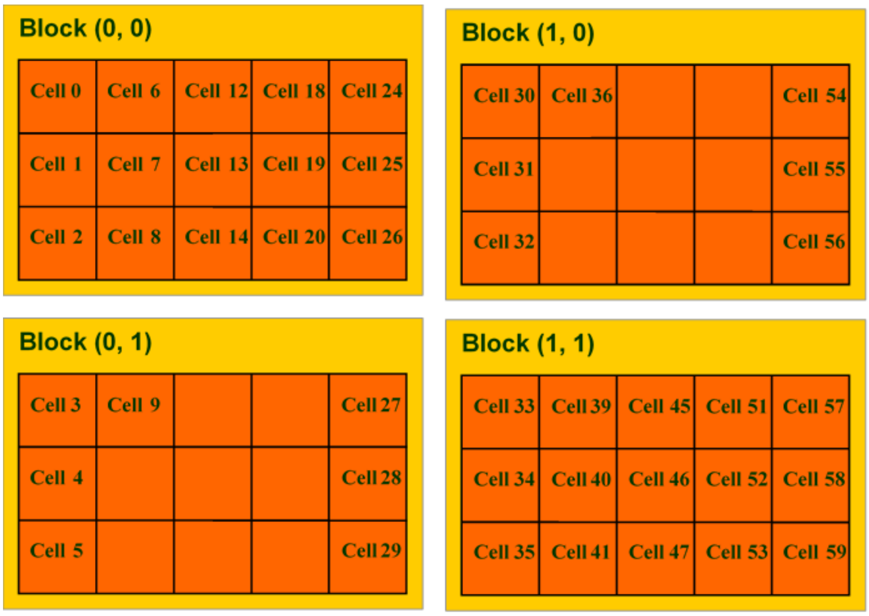
Complete the following program to upload the elements of the Matrix from the host to the GPU device.

```
#define N 16
int main(void) {
  int *a;     // The host copie of the Matrix a
  int *d_a;  //  The device copie of the Matrix a
  int size = N * N * sizeof(int);

  // Allocate space for the host copie of a and setup
input values
  a = (int *)malloc(size); random_ints(a, N * N);


cudaMemcpy(d_a,a,size,cudaMemcpyHostTodevice);
```

We would like to run a kernel on grid configured as M * N matrix of thread blocks. Every thread handles only one cell. Give the statement that calculates the *cell_id* for each thread as shown in the following figure:



Cell_id=((threadIdx.x*gridDim.y*blockDim.y)+(threadidx.y+blockidx.y* blockDim.y))+(blockidx.x*gridDim.y)*(blockDim.x*blockDim.y);

Give the statements that allow to allocate a space memory in the GPU device for a N by N Matrix of integers.

#define N (2048*2048)

Int size = N* sizeof(int);

Int *d_a

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