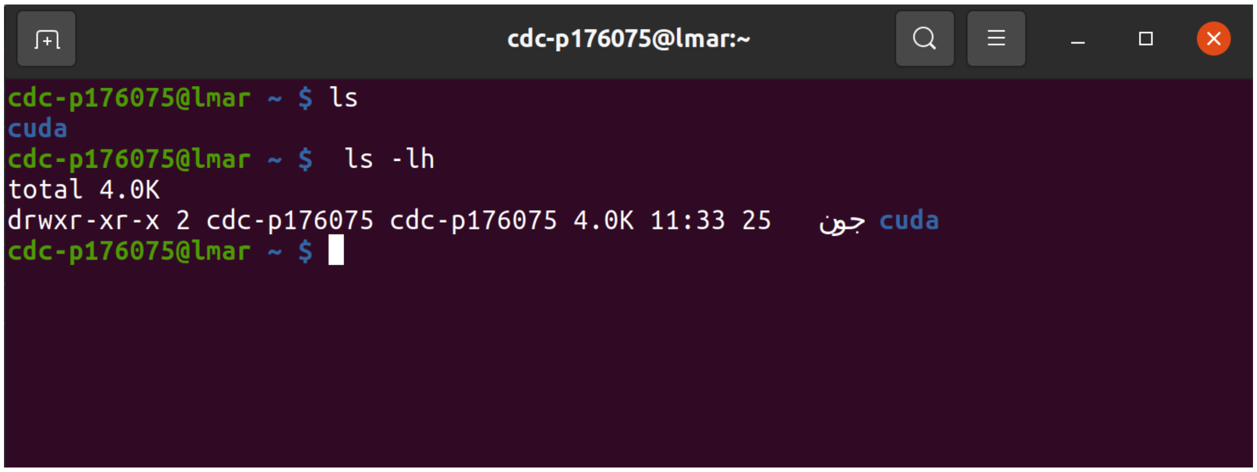
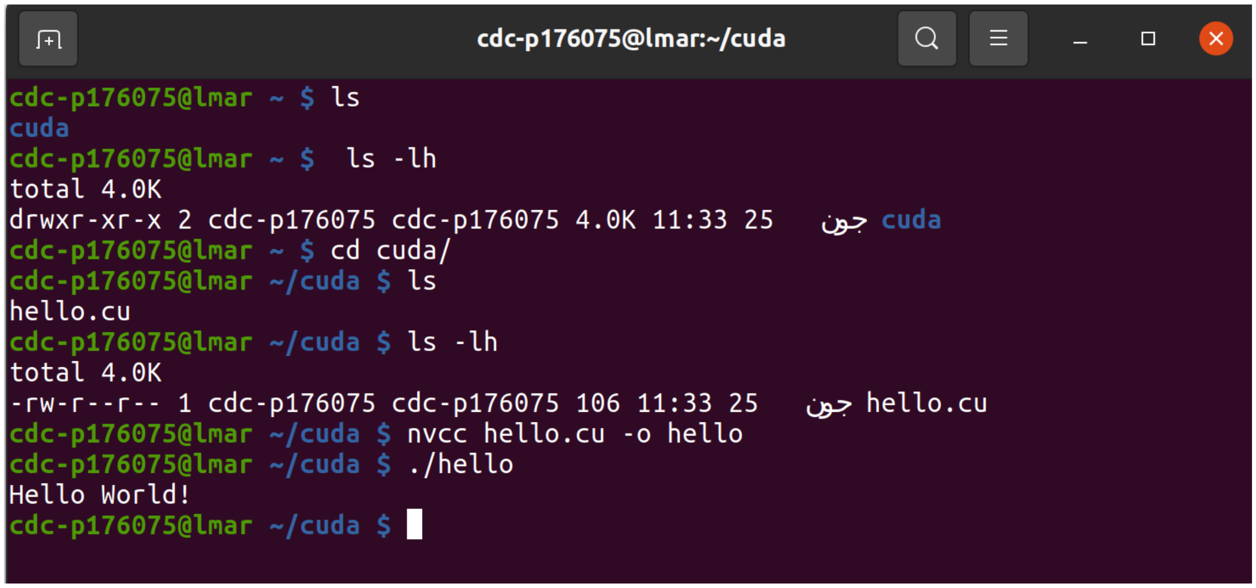
Assignment # 03

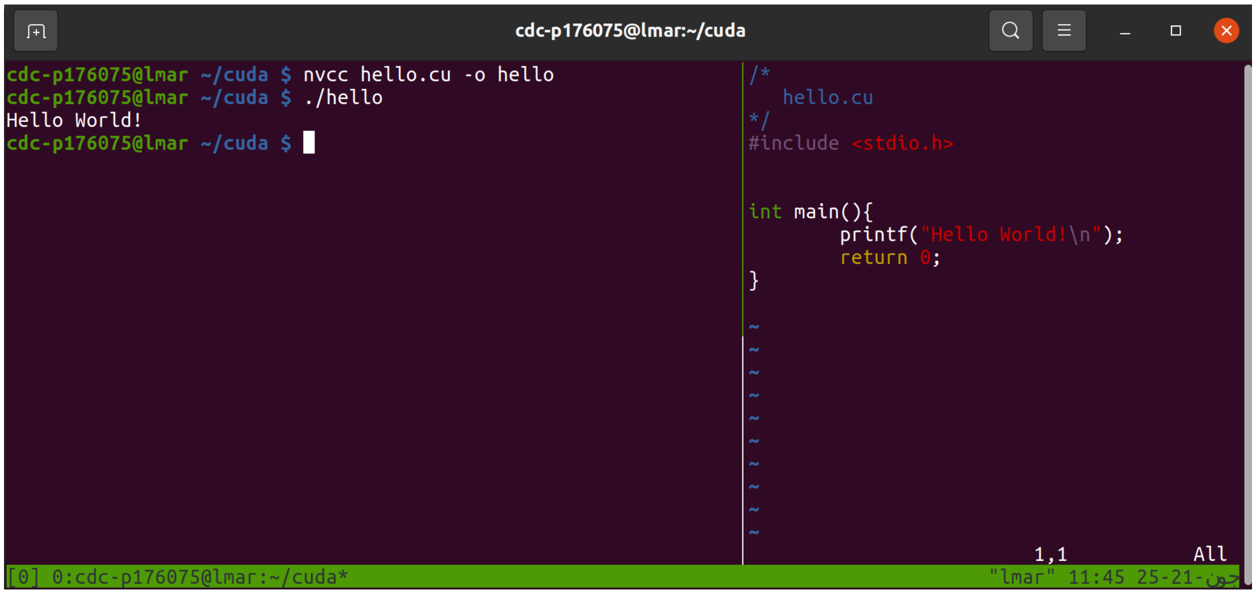
Task # 1:


cdc-p176075@lmar:~
```
cdc-p176075@lmar ~ $ ls
cuda
cdc-p176075@lmar ~ $  ls -lh
total 4.0K
drwxr-xr-x 2 cdc-p176075 cdc-p176075 4.0K 11:33 25   جون cuda
cdc-p176075@lmar ~ $ 
```

Task # 2:


cdc-p176075@lmar:~/cuda
```
cdc-p176075@lmar ~ $ ls
cuda
cdc-p176075@lmar ~ $  ls -lh
total 4.0K
drwxr-xr-x 2 cdc-p176075 cdc-p176075 4.0K 11:33 25   جون cuda
cdc-p176075@lmar ~ $ cd cuda/
cdc-p176075@lmar ~/cuda $ ls
hello.cu
cdc-p176075@lmar ~/cuda $ ls -lh
total 4.0K
-rw-r--r-- 1 cdc-p176075 cdc-p176075 106 11:33 25   جون hello.cu
cdc-p176075@lmar ~/cuda $ nvcc hello.cu -o hello
cdc-p176075@lmar ~/cuda $ ./hello
Hello World!
cdc-p176075@lmar ~/cuda $ 
```

Task # 3:



cdc-p176075@lmar:~/cuda

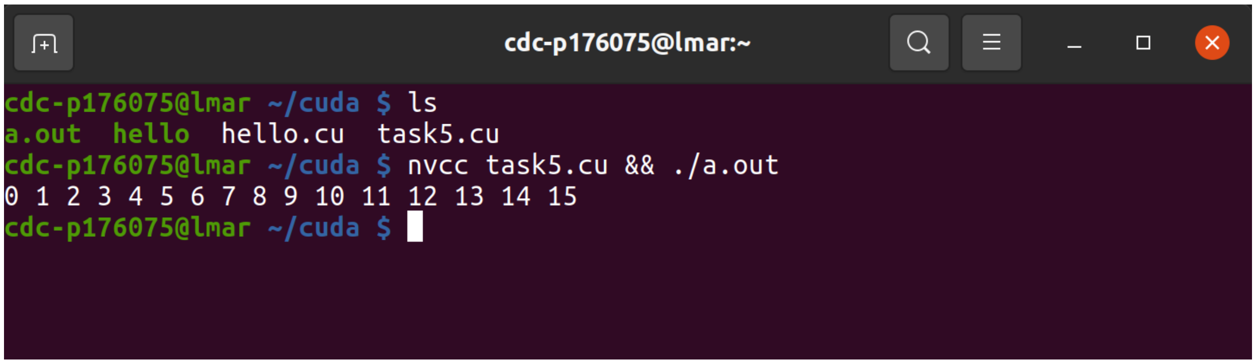
```
cdc-p176075@lmar ~/cuda $ nvcc hello.cu -o hello
cdc-p176075@lmar ~/cuda $ ./hello
Hello World!
cdc-p176075@lmar ~/cuda $
```

```
/*
   hello.cu
*/
#include <stdio.h>


int main(){
        printf("Hello World!\n");
        return 0;
}

~
~
~
~
~
~
~
~
~
~
                                 1,1           All
```

```
[0] 0:cdc-p176075@lmar:~/cuda*                "lmar" 11:45 25-21-جون
```

Task # 4: I have done this, and it helps me alot in case of connection loss.

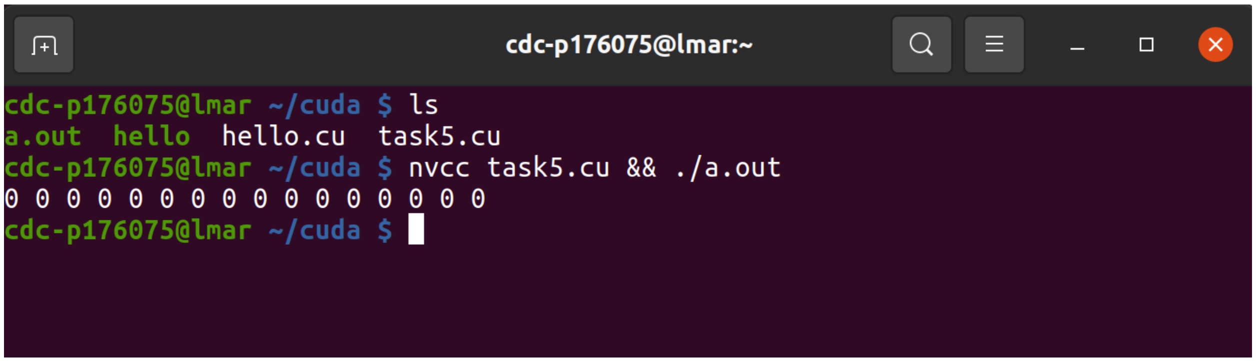
Task # 5a:



cdc-p176075@lmar:~

```
cdc-p176075@lmar ~/cuda $ ls
a.out  hello  hello.cu  task5.cu
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
cdc-p176075@lmar ~/cuda $
```

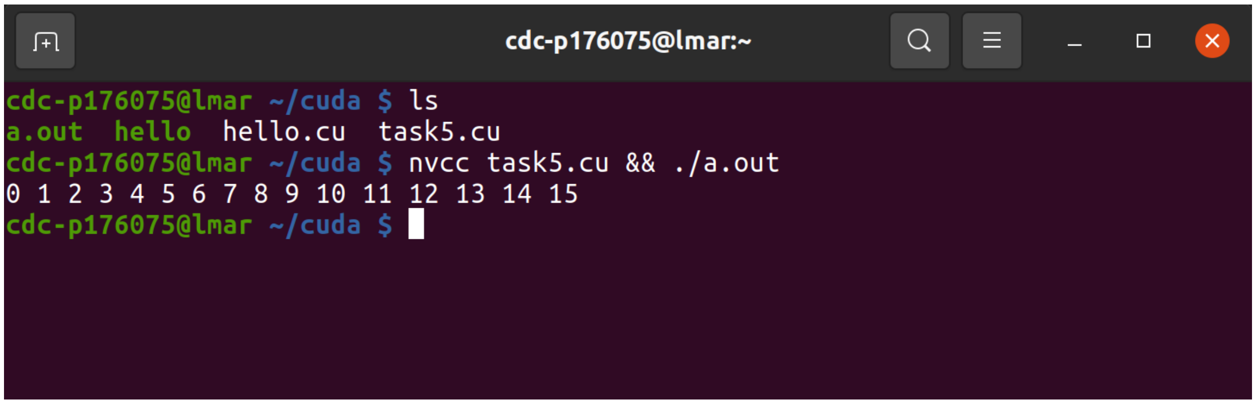
Task # 5b:



cdc-p176075@lmar:~

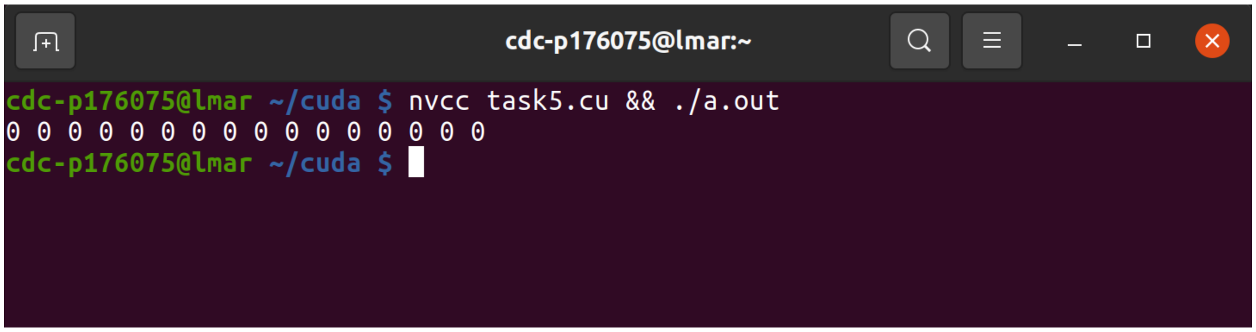
```
cdc-p176075@lmar ~/cuda $ ls
a.out  hello  hello.cu  task5.cu
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
cdc-p176075@lmar ~/cuda $
```

Task # 5c:



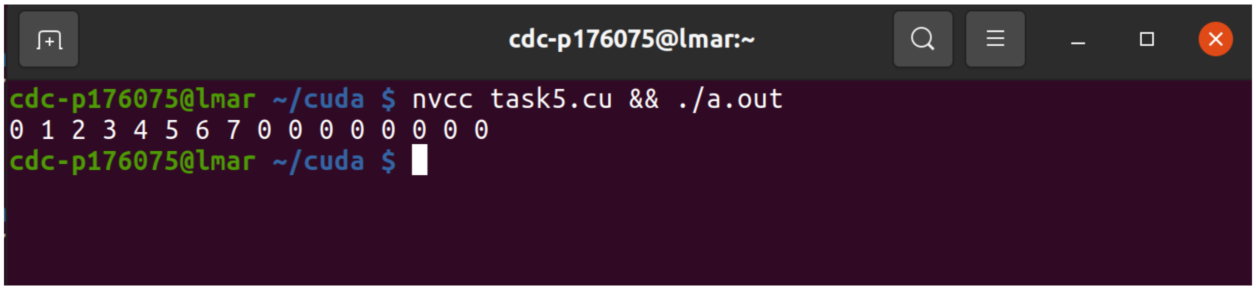
```
cdc-p176075@lmar ~/cuda $ ls
a.out  hello  hello.cu  task5.cu
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
cdc-p176075@lmar ~/cuda $
```

Task # 5d:



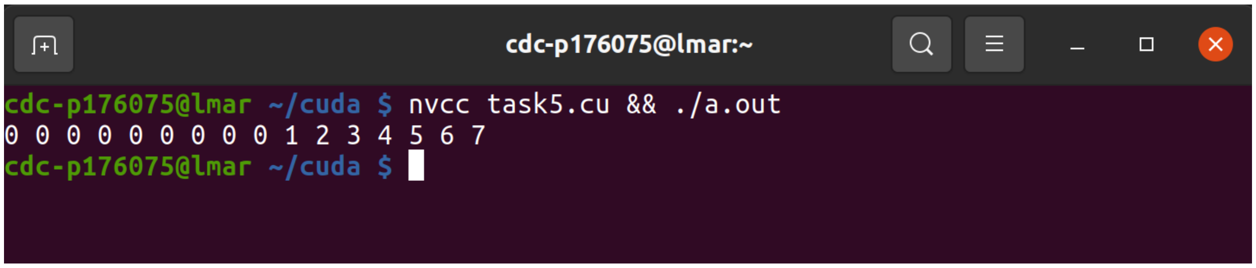
```
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
cdc-p176075@lmar ~/cuda $
```

Task # 5e:



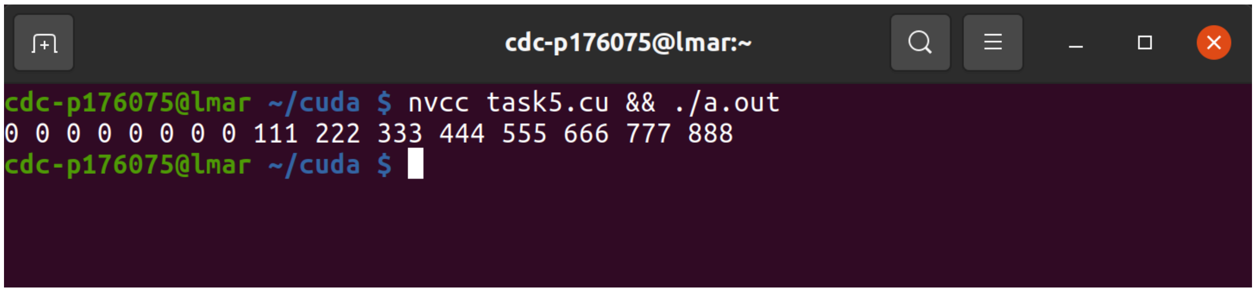
```
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
0 1 2 3 4 5 6 7 0 0 0 0 0 0 0 0
cdc-p176075@lmar ~/cuda $
```

Task # 5f:



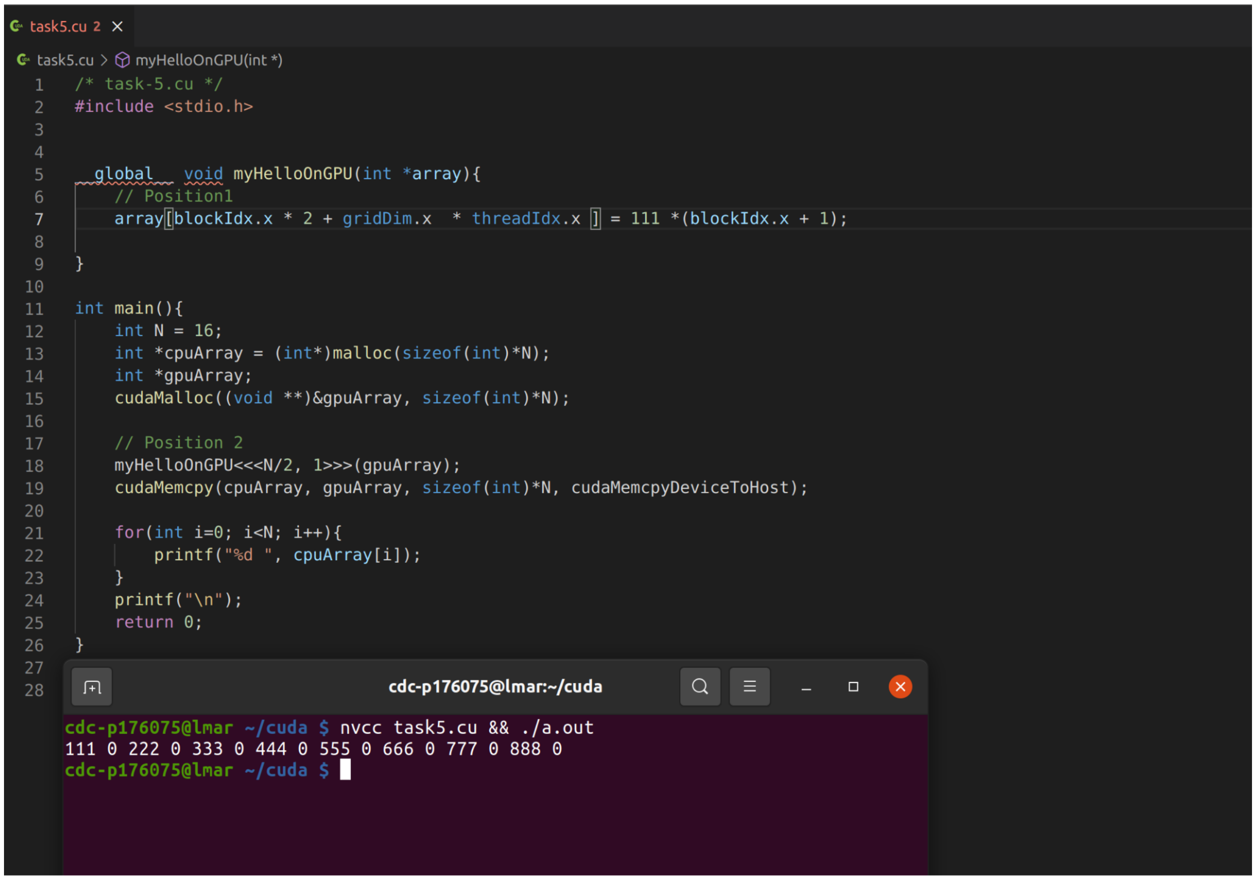
```
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
0 0 0 0 0 0 0 0 0 1 2 3 4 5 6 7
cdc-p176075@lmar ~/cuda $
```

Task # 5g:



```
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
0 0 0 0 0 0 0 0 111 222 333 444 555 666 777 888
cdc-p176075@lmar ~/cuda $ 
```

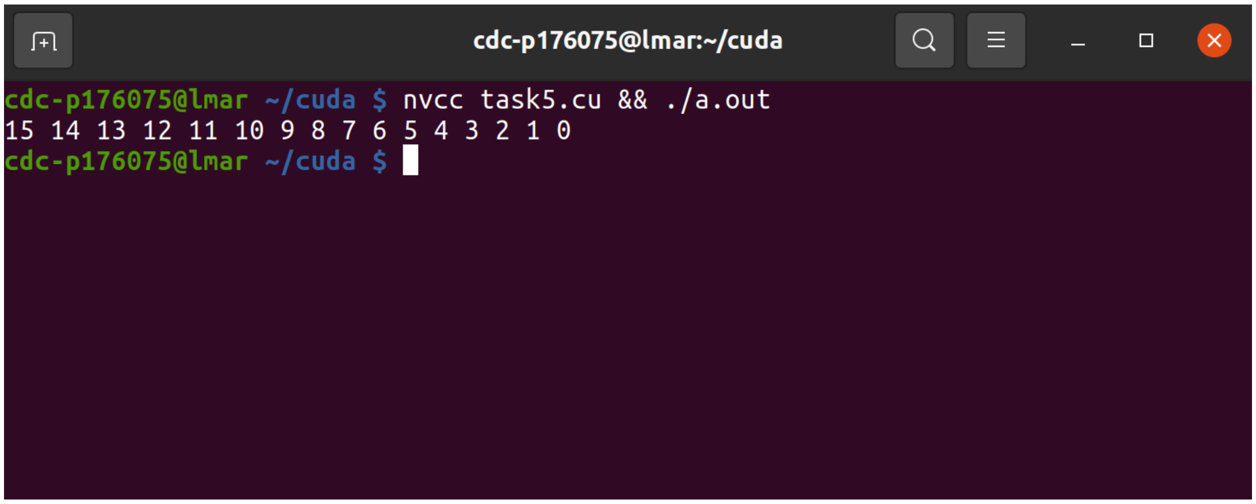
Task # 5h:



```
/* task-5.cu */
#include <stdio.h>


__global__ void myHelloOnGPU(int *array){
    // Position1
    array[blockIdx.x * 2 + gridDim.x  * threadIdx.x ] = 111 *(blockIdx.x + 1);

}

int main(){
    int N = 16;
    int *cpuArray = (int*)malloc(sizeof(int)*N);
    int *gpuArray;
    cudaMalloc((void **)&gpuArray, sizeof(int)*N);

    // Position 2
    myHelloOnGPU<<<N/2, 1>>>(gpuArray);
    cudaMemcpy(cpuArray, gpuArray, sizeof(int)*N, cudaMemcpyDeviceToHost);

    for(int i=0; i<N; i++){
        printf("%d ", cpuArray[i]);
    }
    printf("\n");
    return 0;
}
```

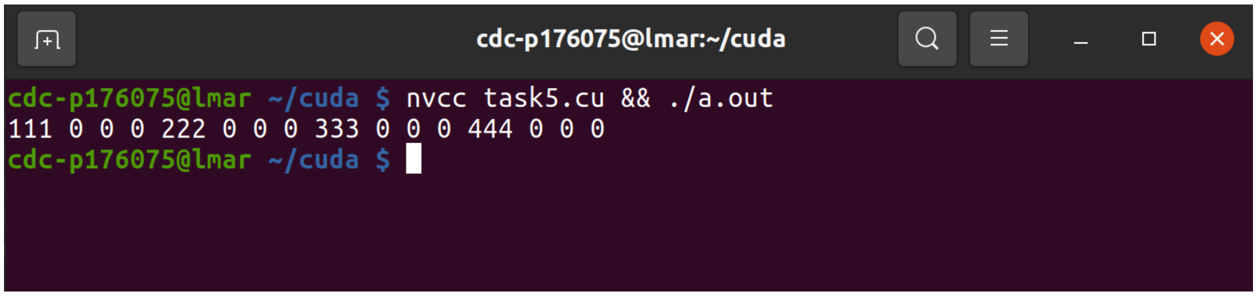
```
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
111 0 222 0 333 0 444 0 555 0 666 0 777 0 888 0
cdc-p176075@lmar ~/cuda $ 
```

Task # 5j:



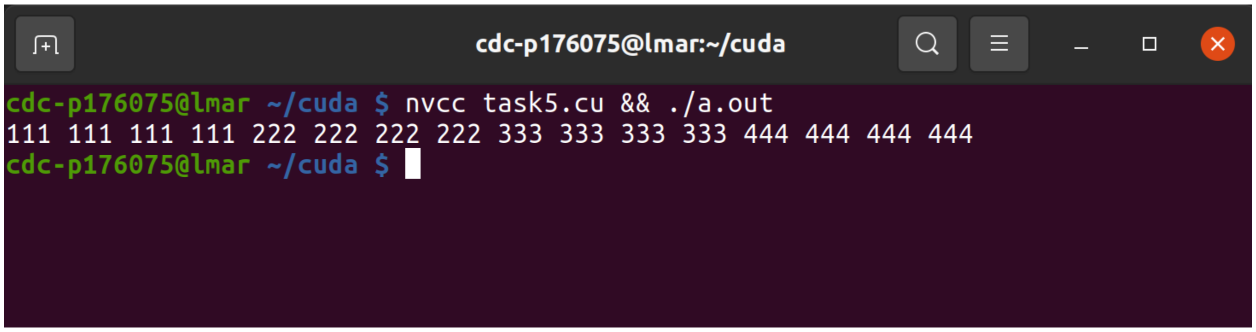
```
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
cdc-p176075@lmar ~/cuda $
```

Task # 5k:



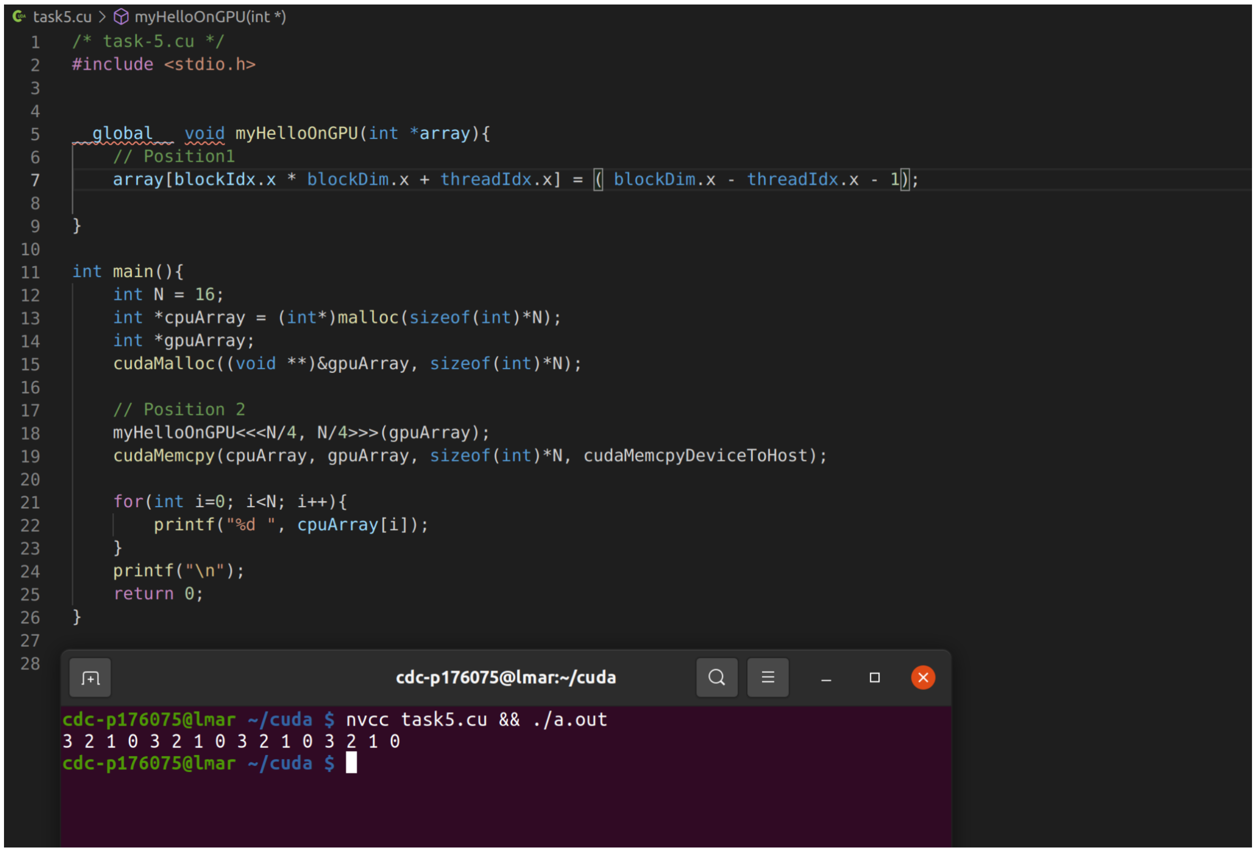
```
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
111 0 0 0 222 0 0 0 333 0 0 0 444 0 0 0
cdc-p176075@lmar ~/cuda $
```

Task # 5m:



```
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
111 111 111 111 222 222 222 222 333 333 333 333 444 444 444 444
cdc-p176075@lmar ~/cuda $
```

Task # 5n:



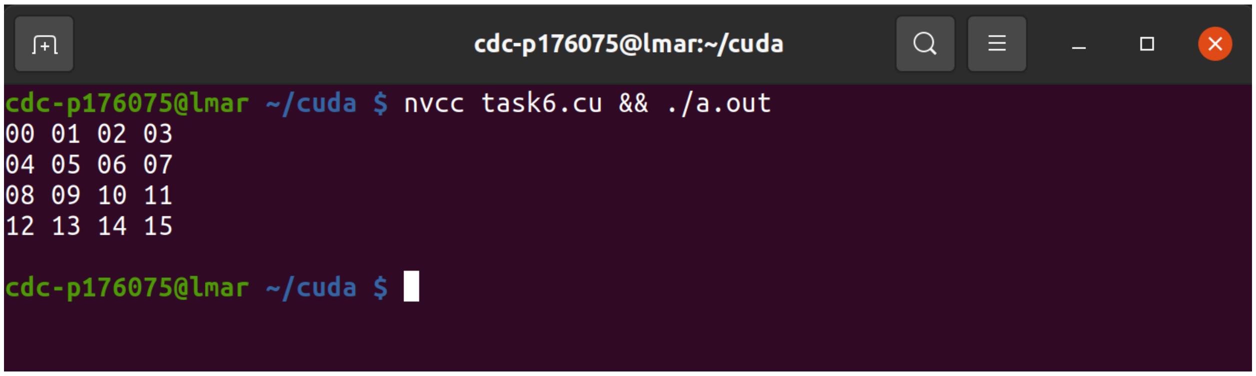
task5.cu > myHelloOnGPU(int *)

```
/* task-5.cu */
#include <stdio.h>


__global__ void myHelloOnGPU(int *array){
    // Position1
    array[blockIdx.x * blockDim.x + threadIdx.x] = ( blockDim.x - threadIdx.x - 1);

}

int main(){
    int N = 16;
    int *cpuArray = (int*)malloc(sizeof(int)*N);
    int *gpuArray;
    cudaMalloc((void **)&gpuArray, sizeof(int)*N);

    // Position 2
    myHelloOnGPU<<<N/4, N/4>>>(gpuArray);
    cudaMemcpy(cpuArray, gpuArray, sizeof(int)*N, cudaMemcpyDeviceToHost);

    for(int i=0; i<N; i++){
        printf("%d ", cpuArray[i]);
    }
    printf("\n");
    return 0;
}
```

cdc-p176075@lmar:~/cuda

```
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0
cdc-p176075@lmar ~/cuda $
```

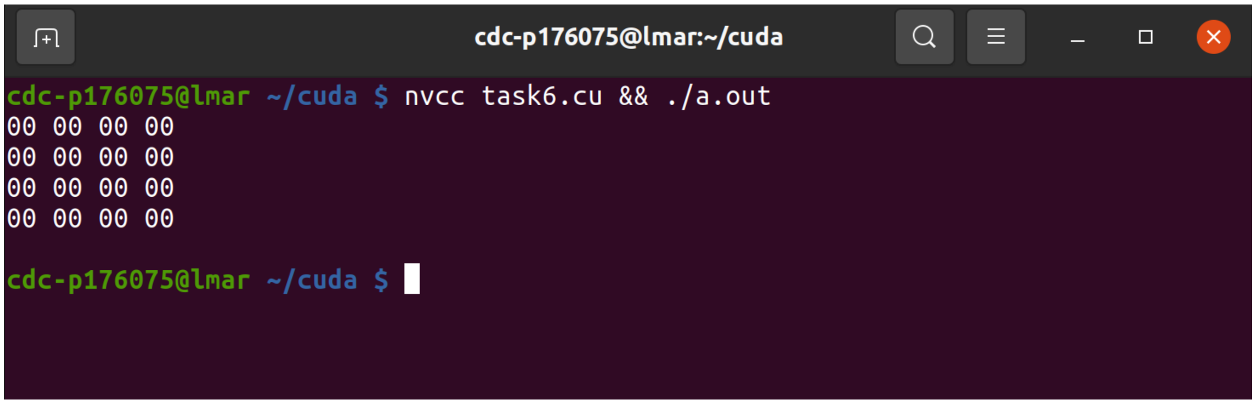
Task # 6a:



cdc-p176075@lmar:~/cuda

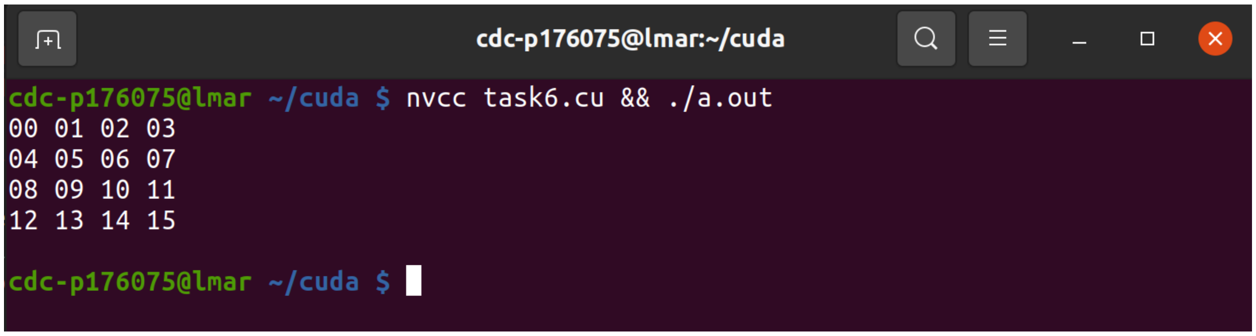
```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
00 01 02 03
04 05 06 07
08 09 10 11
12 13 14 15

cdc-p176075@lmar ~/cuda $
```

Task # 6b:



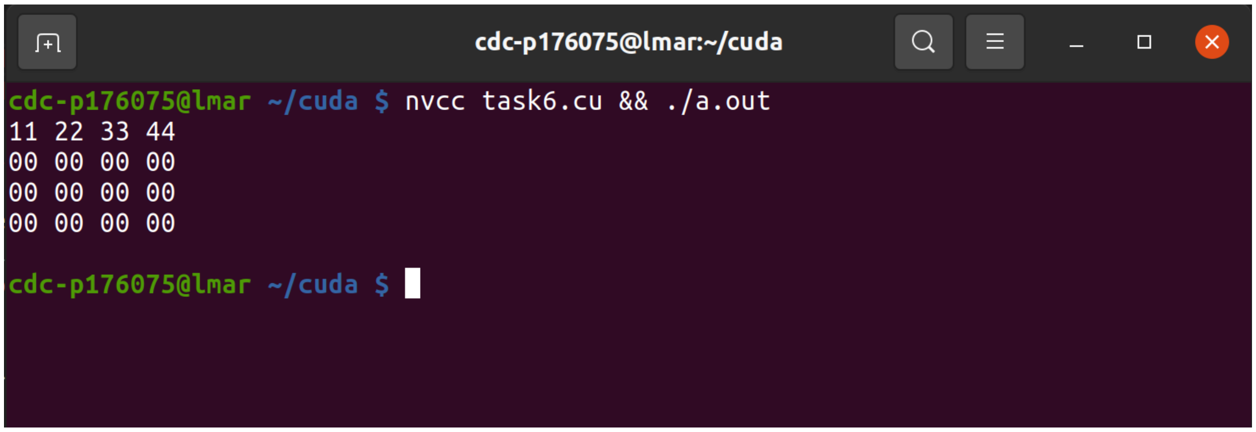
```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00

cdc-p176075@lmar ~/cuda $
```

Task # 6c:



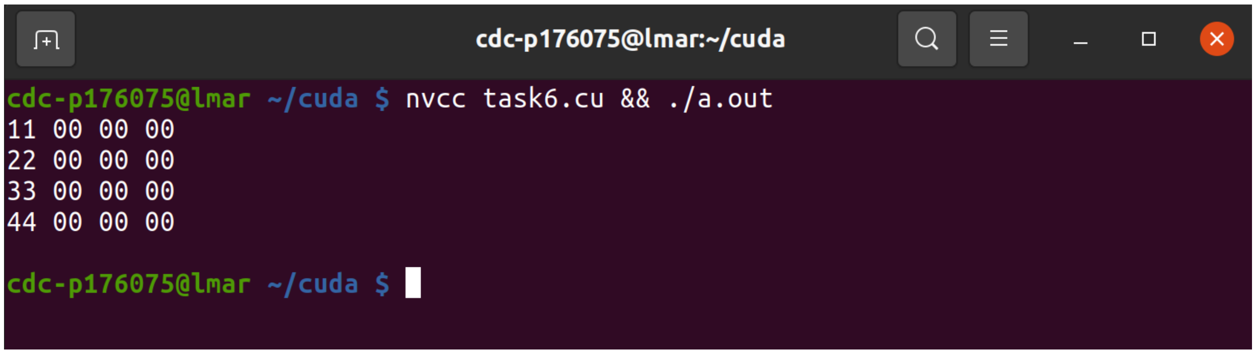
```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
00 01 02 03
04 05 06 07
08 09 10 11
12 13 14 15

cdc-p176075@lmar ~/cuda $
```

Task # 6d:



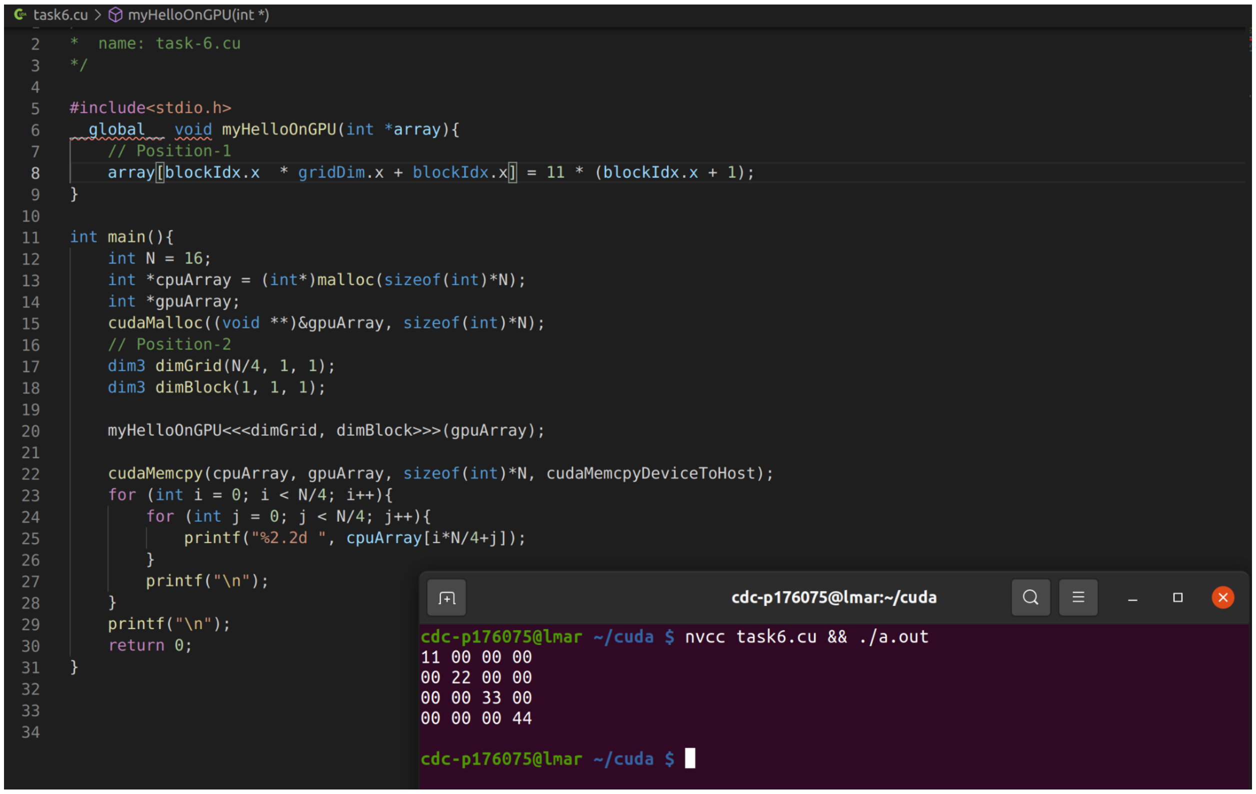
```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
11 22 33 44
00 00 00 00
00 00 00 00
00 00 00 00

cdc-p176075@lmar ~/cuda $
```

Task # 6e:



```
cdc-p176075@lmar:~/cuda
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
11 00 00 00
22 00 00 00
33 00 00 00
44 00 00 00

cdc-p176075@lmar ~/cuda $
```

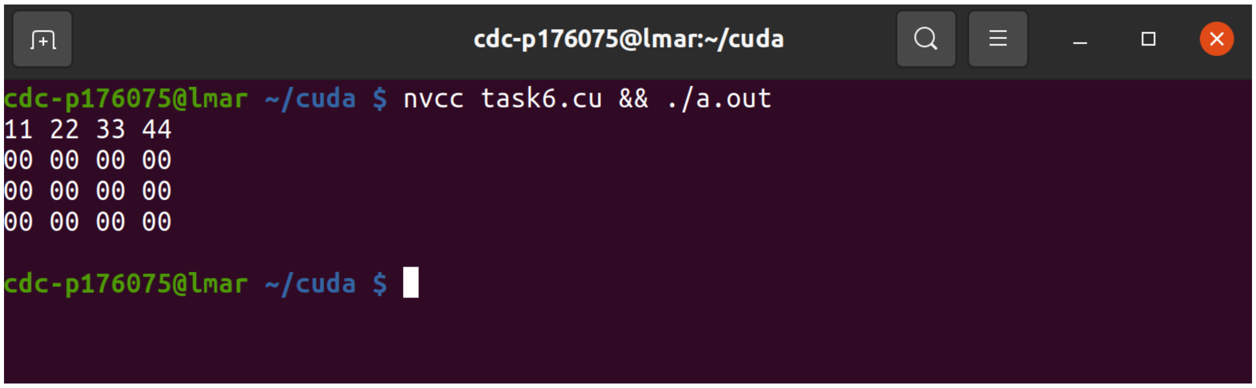
Task # 6f:



```
task6.cu > myHelloOnGPU(int *)
 *  name: task-6.cu
 */

#include<stdio.h>
__global__ void myHelloOnGPU(int *array){
    // Position-1
    array[blockIdx.x  * gridDim.x + blockIdx.x] = 11 * (blockIdx.x + 1);
}

int main(){
    int N = 16;
    int *cpuArray = (int*)malloc(sizeof(int)*N);
    int *gpuArray;
    cudaMalloc((void **)&gpuArray, sizeof(int)*N);
    // Position-2
    dim3 dimGrid(N/4, 1, 1);
    dim3 dimBlock(1, 1, 1);

    myHelloOnGPU<<<dimGrid, dimBlock>>>(gpuArray);

    cudaMemcpy(cpuArray, gpuArray, sizeof(int)*N, cudaMemcpyDeviceToHost);
    for (int i = 0; i < N/4; i++){
        for (int j = 0; j < N/4; j++){
            printf("%2.2d ", cpuArray[i*N/4+j]);
        }
        printf("\n");
    }
    printf("\n");
    return 0;
}
```

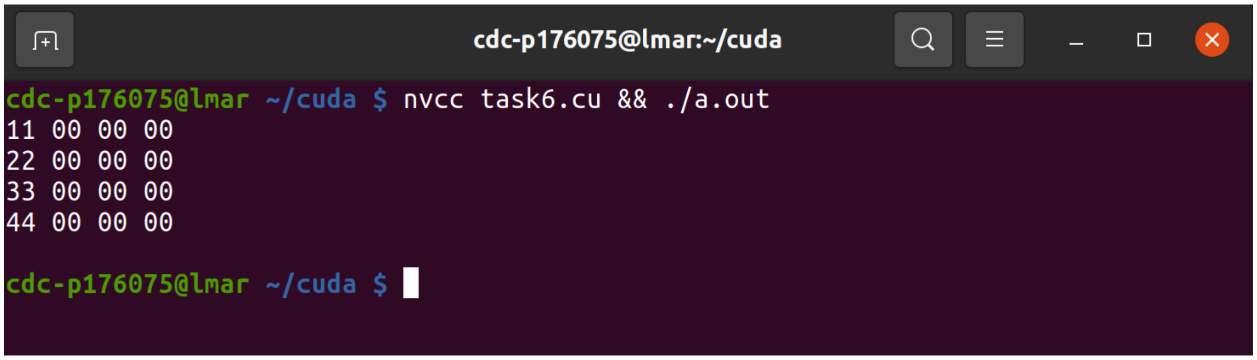
```
cdc-p176075@lmar:~/cuda
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
11 00 00 00
00 22 00 00
00 00 33 00
00 00 00 44

cdc-p176075@lmar ~/cuda $
```

Task # 6g(1):



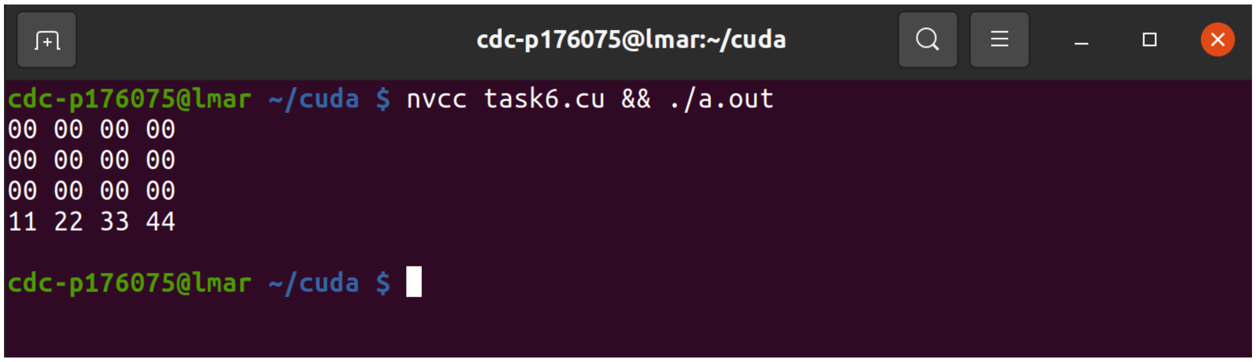
```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
11 22 33 44
00 00 00 00
00 00 00 00
00 00 00 00

cdc-p176075@lmar ~/cuda $
```

Task # 6h(1):



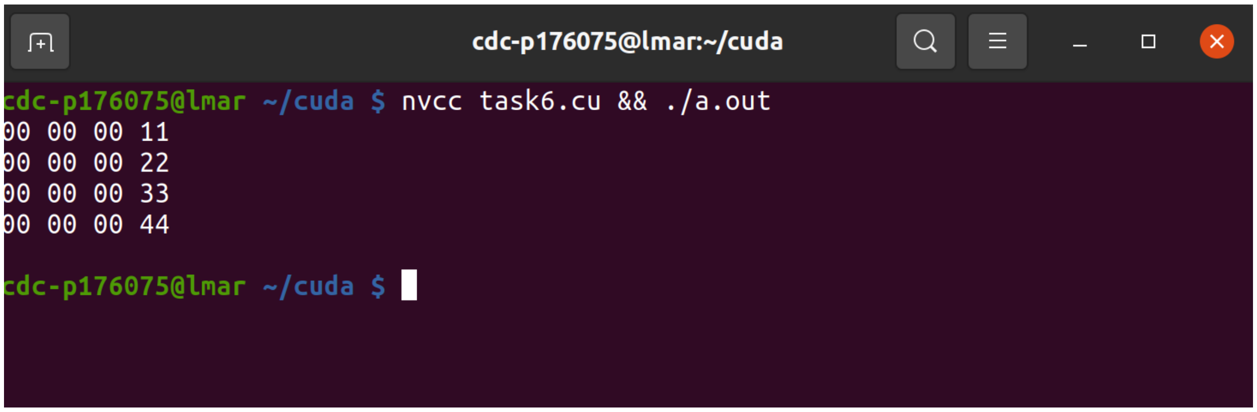
```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
11 00 00 00
22 00 00 00
33 00 00 00
44 00 00 00

cdc-p176075@lmar ~/cuda $
```

Task # 6g(2):



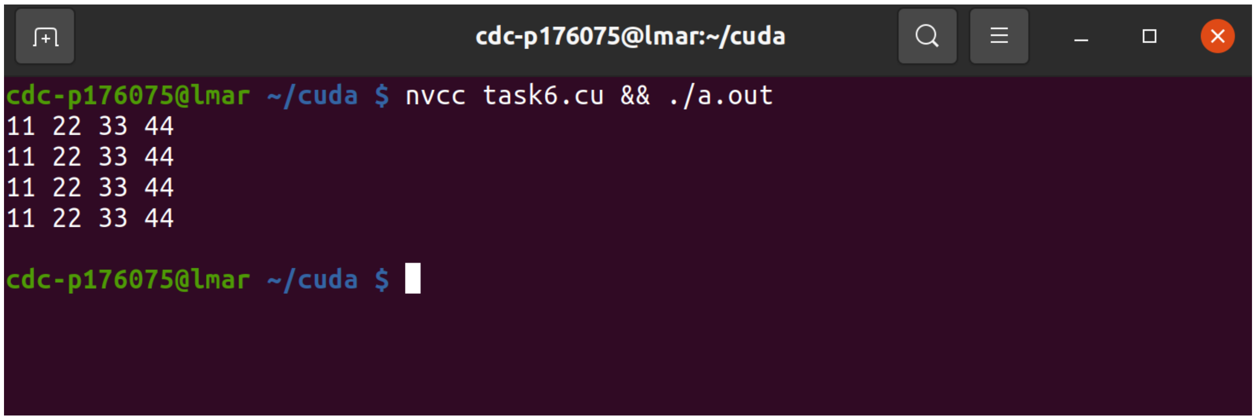
```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
00 00 00 00
00 00 00 00
00 00 00 00
11 22 33 44

cdc-p176075@lmar ~/cuda $
```

Task # 6h(2):



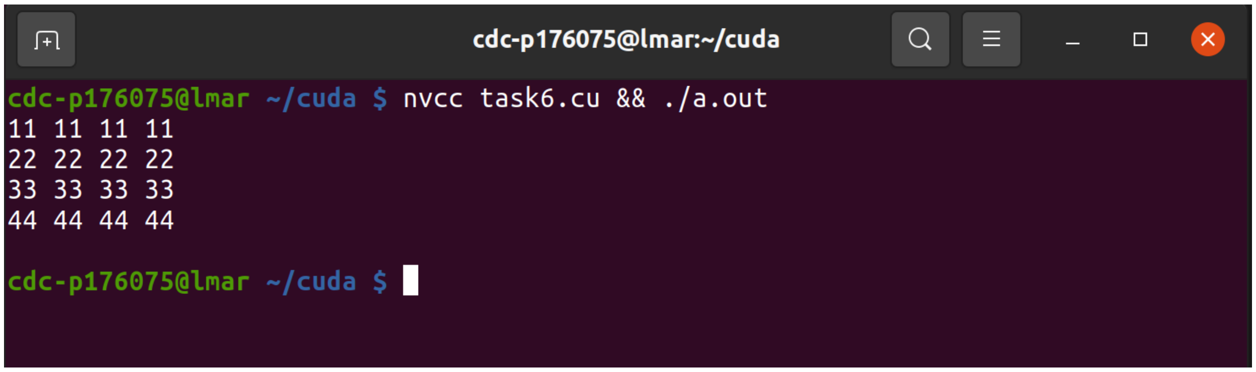
```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
00 00 00 11
00 00 00 22
00 00 00 33
00 00 00 44

cdc-p176075@lmar ~/cuda $
```

Task # 6j:



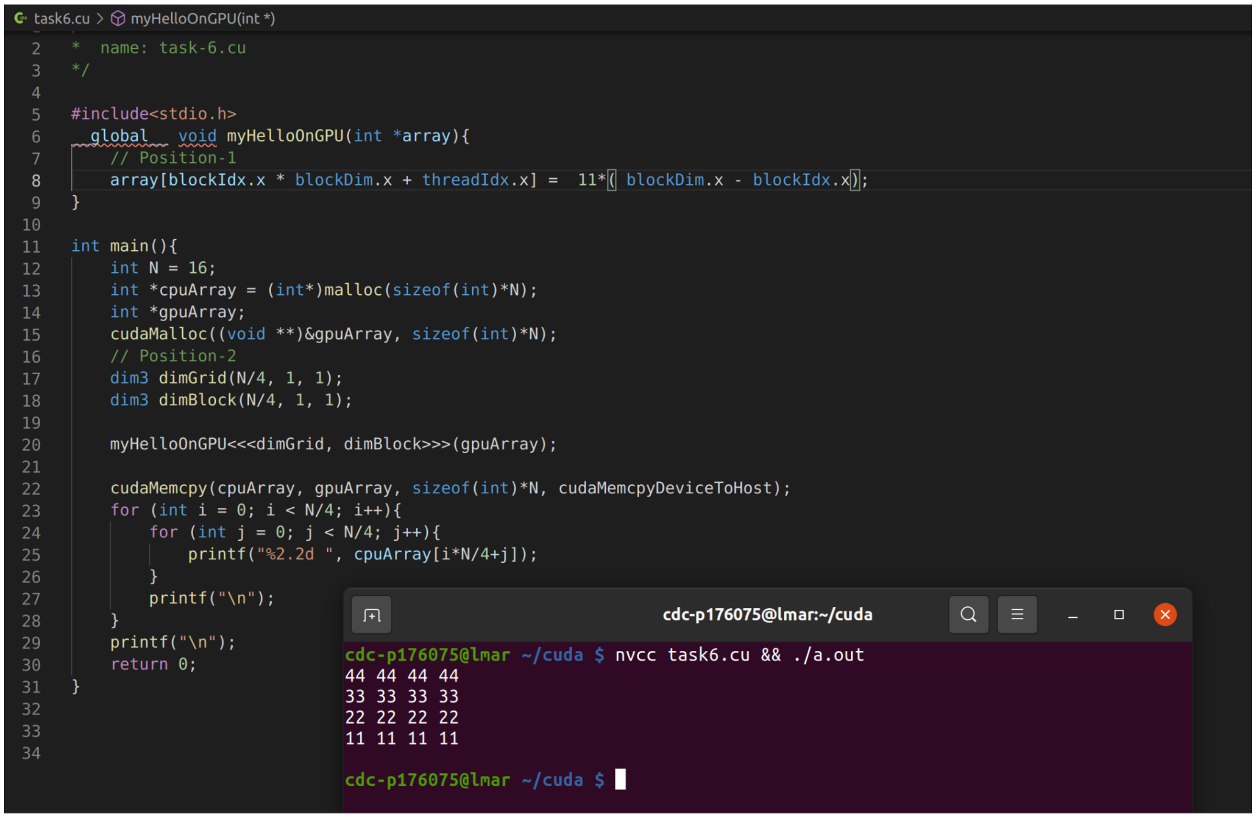
```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
11 22 33 44
11 22 33 44
11 22 33 44
11 22 33 44

cdc-p176075@lmar ~/cuda $
```

Task # 6k:



```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
11 11 11 11
22 22 22 22
33 33 33 33
44 44 44 44

cdc-p176075@lmar ~/cuda $
```

Task # 6m:



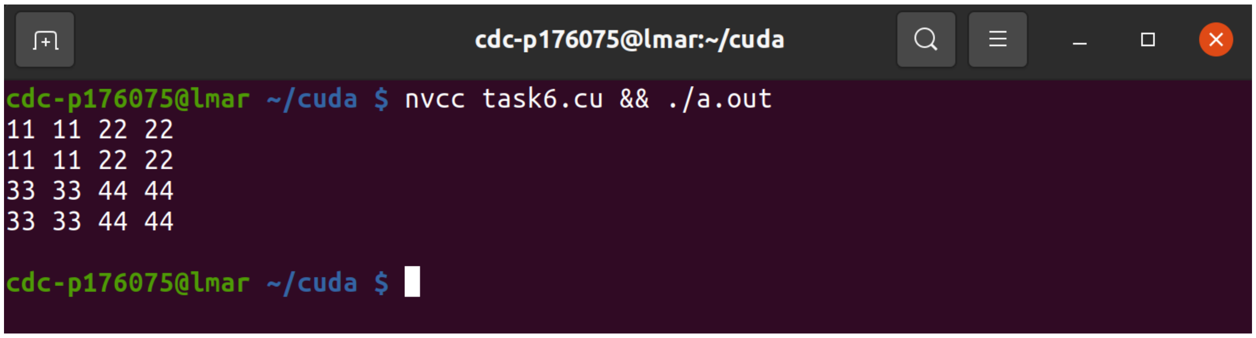
task6.cu > myHelloOnGPU(int *)

```
 *  name: task-6.cu
 */

#include<stdio.h>
__global__ void myHelloOnGPU(int *array){
    // Position-1
    array[blockIdx.x * blockDim.x + threadIdx.x] =  11*( blockDim.x - blockIdx.x);
}

int main(){
    int N = 16;
    int *cpuArray = (int*)malloc(sizeof(int)*N);
    int *gpuArray;
    cudaMalloc((void **)&gpuArray, sizeof(int)*N);
    // Position-2
    dim3 dimGrid(N/4, 1, 1);
    dim3 dimBlock(N/4, 1, 1);

    myHelloOnGPU<<<dimGrid, dimBlock>>>(gpuArray);

    cudaMemcpy(cpuArray, gpuArray, sizeof(int)*N, cudaMemcpyDeviceToHost);
    for (int i = 0; i < N/4; i++){
        for (int j = 0; j < N/4; j++){
            printf("%2.2d ", cpuArray[i*N/4+j]);
        }
        printf("\n");
    }
    printf("\n");
    return 0;
}
```

cdc-p176075@lmar:~/cuda

```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
44 44 44 44
33 33 33 33
22 22 22 22
11 11 11 11

cdc-p176075@lmar ~/cuda $
```

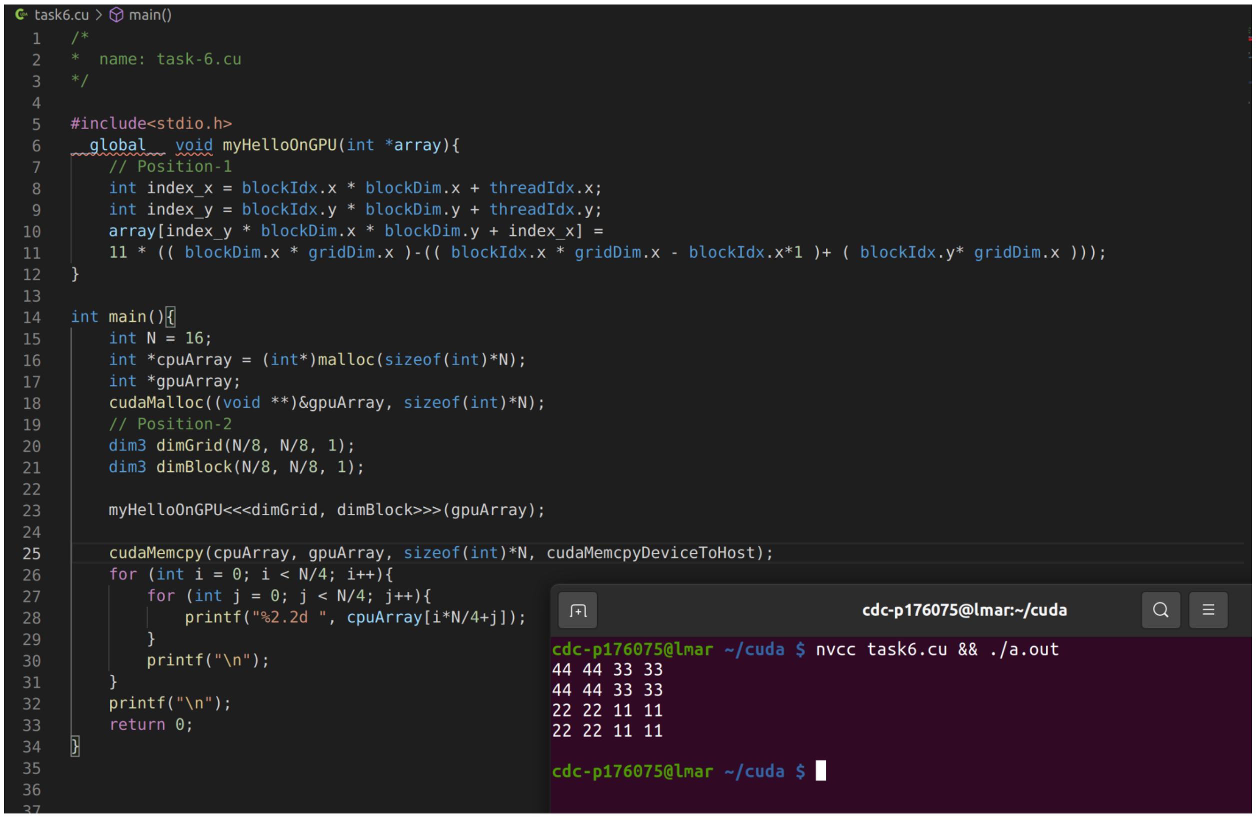
Task # 6n:



cdc-p176075@lmar:~/cuda

```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
11 11 22 22
11 11 22 22
33 33 44 44
33 33 44 44

cdc-p176075@lmar ~/cuda $
```

Task # 6o:



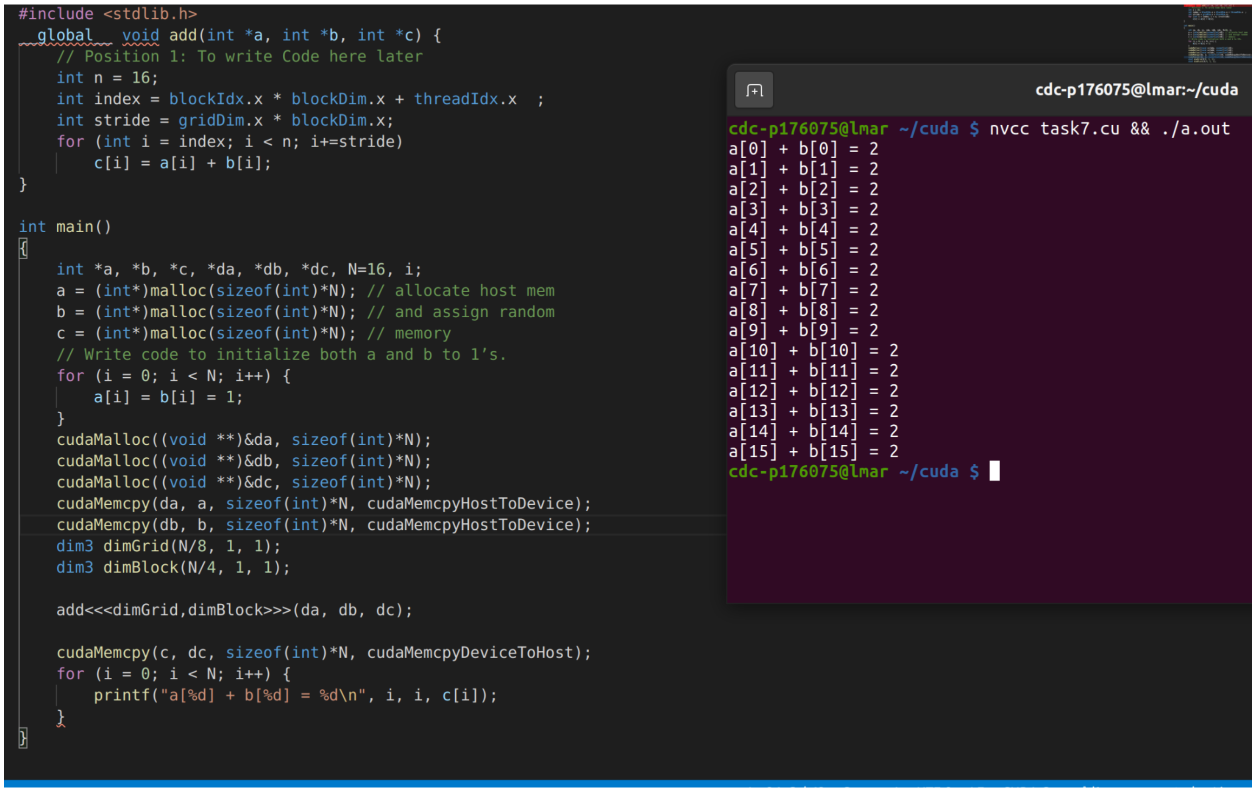
task6.cu > main()

```
/*
*  name: task-6.cu
*/

#include<stdio.h>
__global__ void myHelloOnGPU(int *array){
    // Position-1
    int index_x = blockIdx.x * blockDim.x + threadIdx.x;
    int index_y = blockIdx.y * blockDim.y + threadIdx.y;
    array[index_y * blockDim.x * blockDim.y + index_x] =
    11 * (( blockDim.x * gridDim.x )-(( blockIdx.x * gridDim.x - blockIdx.x*1 )+ ( blockIdx.y* gridDim.x )));
}

int main(){
    int N = 16;
    int *cpuArray = (int*)malloc(sizeof(int)*N);
    int *gpuArray;
    cudaMalloc((void **)&gpuArray, sizeof(int)*N);
    // Position-2
    dim3 dimGrid(N/8, N/8, 1);
    dim3 dimBlock(N/8, N/8, 1);

    myHelloOnGPU<<<dimGrid, dimBlock>>>(gpuArray);

    cudaMemcpy(cpuArray, gpuArray, sizeof(int)*N, cudaMemcpyDeviceToHost);
    for (int i = 0; i < N/4; i++){
        for (int j = 0; j < N/4; j++){
            printf("%2.2d ", cpuArray[i*N/4+j]);
        }
        printf("\n");
    }
    printf("\n");
    return 0;
}
```

cdc-p176075@lmar:~/cuda

```
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
44 44 33 33
44 44 33 33
22 22 11 11
22 22 11 11

cdc-p176075@lmar ~/cuda $
```

Task # 7:

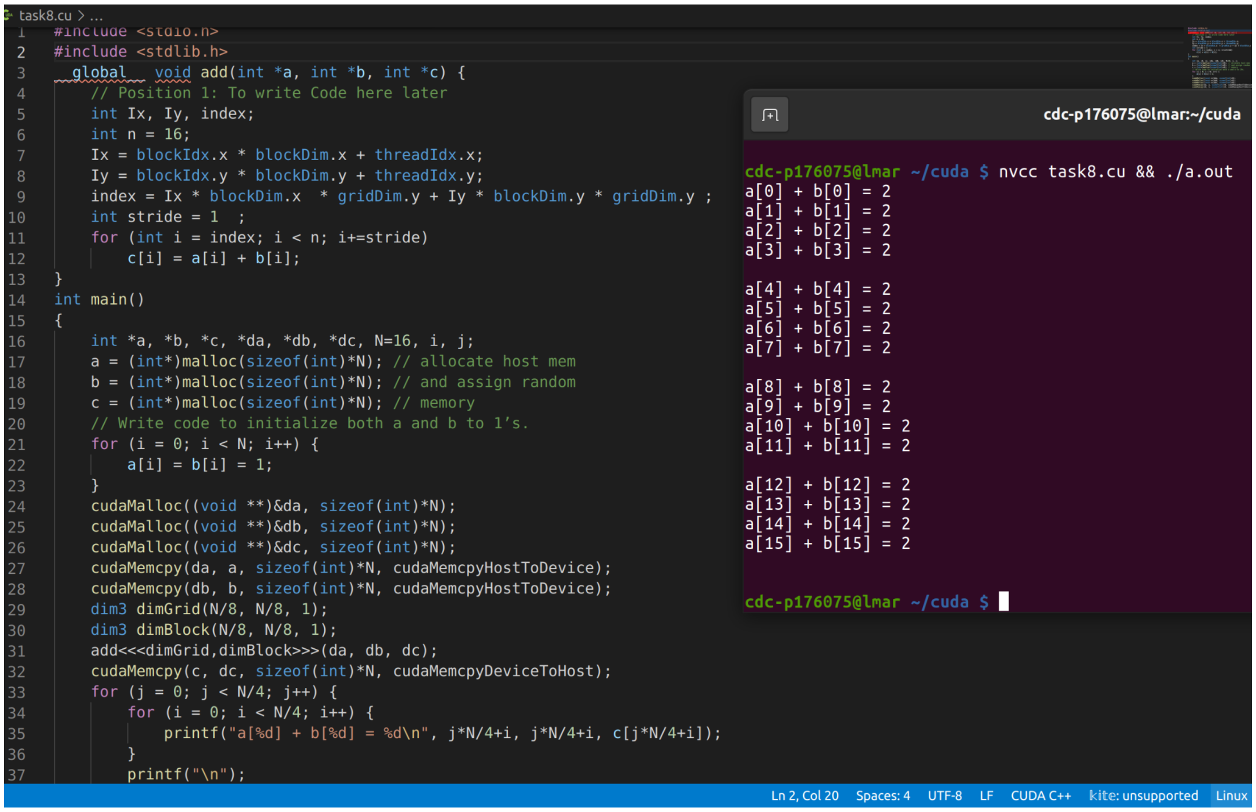


```
#include <stdlib.h>
__global__ void add(int *a, int *b, int *c) {
    // Position 1: To write Code here later
    int n = 16;
    int index = blockIdx.x * blockDim.x + threadIdx.x  ;
    int stride = gridDim.x * blockDim.x;
    for (int i = index; i < n; i+=stride)
        c[i] = a[i] + b[i];
}

int main()
{
    int *a, *b, *c, *da, *db, *dc, N=16, i;
    a = (int*)malloc(sizeof(int)*N); // allocate host mem
    b = (int*)malloc(sizeof(int)*N); // and assign random
    c = (int*)malloc(sizeof(int)*N); // memory
    // Write code to initialize both a and b to 1's.
    for (i = 0; i < N; i++) {
        a[i] = b[i] = 1;
    }
    cudaMalloc((void **)&da, sizeof(int)*N);
    cudaMalloc((void **)&db, sizeof(int)*N);
    cudaMalloc((void **)&dc, sizeof(int)*N);
    cudaMemcpy(da, a, sizeof(int)*N, cudaMemcpyHostToDevice);
    cudaMemcpy(db, b, sizeof(int)*N, cudaMemcpyHostToDevice);
    dim3 dimGrid(N/8, 1, 1);
    dim3 dimBlock(N/4, 1, 1);

    add<<<dimGrid,dimBlock>>>(da, db, dc);

    cudaMemcpy(c, dc, sizeof(int)*N, cudaMemcpyDeviceToHost);
    for (i = 0; i < N; i++) {
        printf("a[%d] + b[%d] = %d\n", i, i, c[i]);
    }
}
```

cdc-p176075@lmar:~/cuda

```
cdc-p176075@lmar ~/cuda $ nvcc task7.cu && ./a.out
a[0] + b[0] = 2
a[1] + b[1] = 2
a[2] + b[2] = 2
a[3] + b[3] = 2
a[4] + b[4] = 2
a[5] + b[5] = 2
a[6] + b[6] = 2
a[7] + b[7] = 2
a[8] + b[8] = 2
a[9] + b[9] = 2
a[10] + b[10] = 2
a[11] + b[11] = 2
a[12] + b[12] = 2
a[13] + b[13] = 2
a[14] + b[14] = 2
a[15] + b[15] = 2
cdc-p176075@lmar ~/cuda $
```

Task # 8:



task8.cu > ...

```
#include <stdio.h>
#include <stdlib.h>
__global__ void add(int *a, int *b, int *c) {
    // Position 1: To write Code here later
    int Ix, Iy, index;
    int n = 16;
    Ix = blockIdx.x * blockDim.x + threadIdx.x;
    Iy = blockIdx.y * blockDim.y + threadIdx.y;
    index = Ix * blockDim.x  * gridDim.y + Iy * blockDim.y * gridDim.y ;
    int stride = 1  ;
    for (int i = index; i < n; i+=stride)
        c[i] = a[i] + b[i];
}
int main()
{
    int *a, *b, *c, *da, *db, *dc, N=16, i, j;
    a = (int*)malloc(sizeof(int)*N); // allocate host mem
    b = (int*)malloc(sizeof(int)*N); // and assign random
    c = (int*)malloc(sizeof(int)*N); // memory
    // Write code to initialize both a and b to 1's.
    for (i = 0; i < N; i++) {
        a[i] = b[i] = 1;
    }
    cudaMalloc((void **)&da, sizeof(int)*N);
    cudaMalloc((void **)&db, sizeof(int)*N);
    cudaMalloc((void **)&dc, sizeof(int)*N);
    cudaMemcpy(da, a, sizeof(int)*N, cudaMemcpyHostToDevice);
    cudaMemcpy(db, b, sizeof(int)*N, cudaMemcpyHostToDevice);
    dim3 dimGrid(N/8, N/8, 1);
    dim3 dimBlock(N/8, N/8, 1);
    add<<<dimGrid,dimBlock>>>(da, db, dc);
    cudaMemcpy(c, dc, sizeof(int)*N, cudaMemcpyDeviceToHost);
    for (j = 0; j < N/4; j++) {
        for (i = 0; i < N/4; i++) {
            printf("a[%d] + b[%d] = %d\n", j*N/4+i, j*N/4+i, c[j*N/4+i]);
        }
        printf("\n");
```

Ln 2, Col 20   Spaces: 4   UTF-8   LF   CUDA C++   kite: unsupported   Linux

cdc-p176075@lmar:~/cuda

```
cdc-p176075@lmar ~/cuda $ nvcc task8.cu && ./a.out
a[0] + b[0] = 2
a[1] + b[1] = 2
a[2] + b[2] = 2
a[3] + b[3] = 2

a[4] + b[4] = 2
a[5] + b[5] = 2
a[6] + b[6] = 2
a[7] + b[7] = 2

a[8] + b[8] = 2
a[9] + b[9] = 2
a[10] + b[10] = 2
a[11] + b[11] = 2

a[12] + b[12] = 2
a[13] + b[13] = 2
a[14] + b[14] = 2
a[15] + b[15] = 2


cdc-p176075@lmar ~/cuda $ 
```