National University
Of Computer and Emerging Sciences

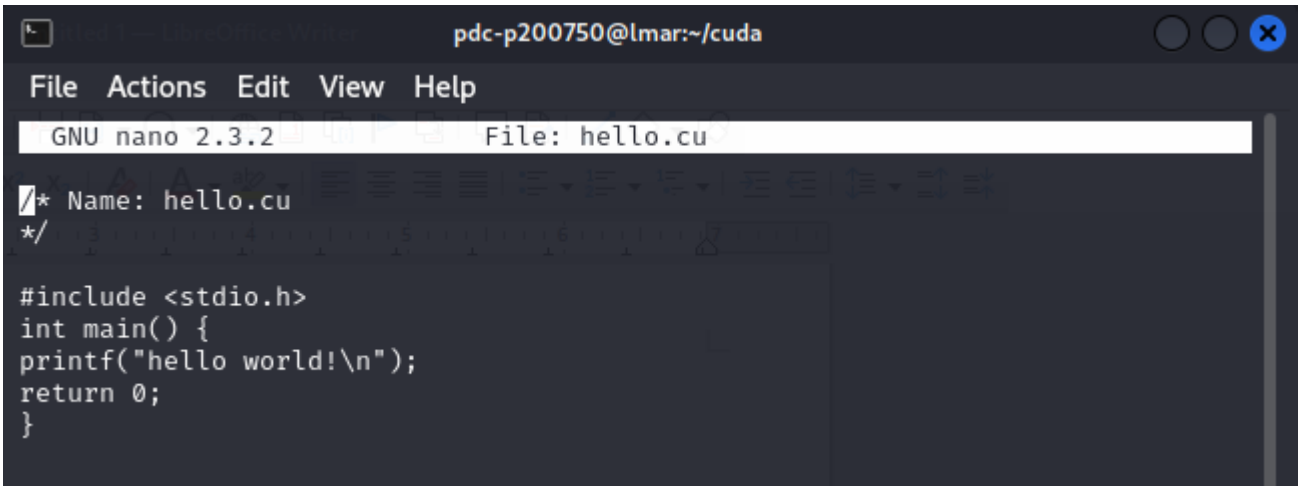
Name : Waqar Ahmed

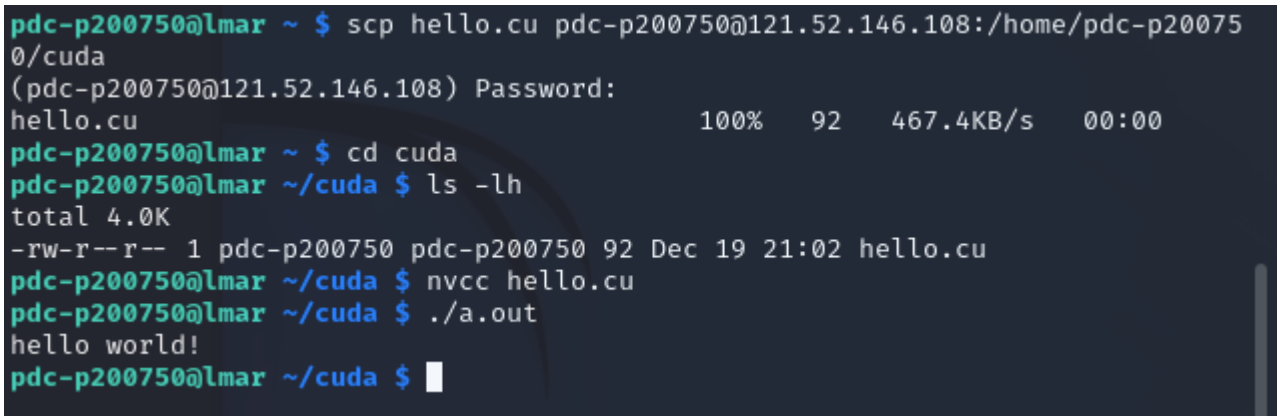
Roll No: 20P-0750

Assignment #02

Task 1 : Login successfully

Task 2: Hello World





Task 5: Playing with 1D GPU indices

1. Task 5a

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
pdc-p200750@lmar ~/cuda $
```

2. Task 5b

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
pdc-p200750@lmar ~/cuda $
```

3.Task 5c

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
pdc-p200750@lmar ~/cuda $
```

4. Task 5d

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
pdc-p200750@lmar ~/cuda $
```

5. Task 5e

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
0 1 2 3 4 5 6 7 0 0 0 0 0 0 0 0
pdc-p200750@lmar ~/cuda $
```

6. Task 5f

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
0 0 0 0 0 0 0 0 1 2 3 4 5 6 7
pdc-p200750@lmar ~/cuda $
```

7. Task 5g

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
0 0 0 0 0 0 0 0 111 222 333 444 555 666 777 888
pdc-p200750@lmar ~/cuda $
```

8. Task 5h: What should be Position 1 and 2 in order to obtain the following output:

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
111 0 222 0 333 0 444 0 555 0 666 0 777 0 888 0
```

Position 2: myHelloOnGPU<<<N/2, 1>>>(gpuArray);

Position 1: array[blockIdx.x * 2] = 111 *(blockIdx.x + 1);

9. Task 5j

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

10. Task 5k

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
111 0 0 0 222 0 0 0 333 0 0 0 444 0 0 0
pdc-p200750@lmar ~/cuda $
```

11. Task 5m

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
111 111 111 111 222 222 222 222 333 333 333 333 444 444 444 444
pdc-p200750@lmar ~/cuda $
```

12. Task 5n: What should be Position 1 and 2 in order to obtain the following output:

```
pdc-p200750@lmar ~/cuda $ nvcc task5.c
pdc-p200750@lmar ~/cuda $ ./a.out
3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0
pdc-p200750@lmar ~/cuda $
```

Position 2: myHelloOnGPU<<<N/4, N/4>>>(gpuArray);

Position 1: array[blockIdx.x * blockDim.x + threadIdx.x] = blockDim.x – threadIdx.x – 1;

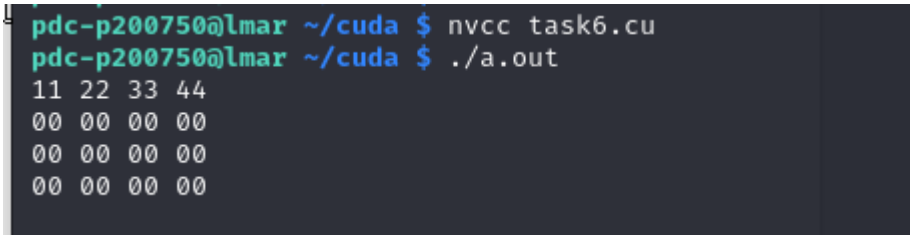
Task 6: Playing with 2D GPU indices.

1. Task 6a:

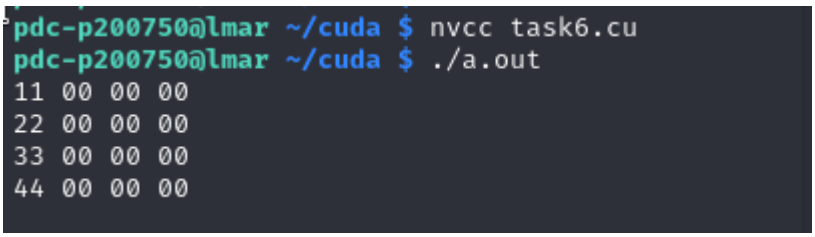
2. Task 6b:

3. Task 6c:

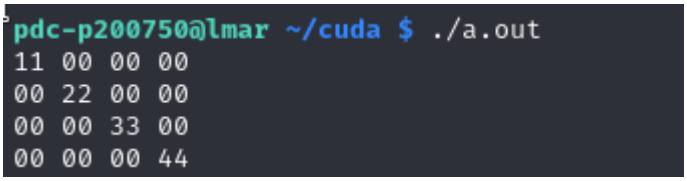
4. Task 6d:



5. Task 6e:



6. Task 6f:



for above output we need to change in position 1 and position 2 as following.

Position1 : int index = threadIdx.x * blockDim.x;

array[index] = (index % 5== 0) ? (index / 5+1)* 11: 0;

Position2: dim3 dimGrid(4,1,1); dim3 blockDim(4,1,1);

7. Task 6g:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
00 00 00 00
00 00 00 00
00 00 00 00
11 22 33 44
```

8. Task 6f:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
11 00 00 00
22 00 00 00
33 00 00 00
44 00 00 00
```

9. Task 6g:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
00 00 00 00
00 00 00 00
00 00 00 00
11 22 33 44
```

10. Task 6h:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
00 00 00 11
00 00 00 22
00 00 00 33
00 00 00 44
```

11. Task 6j:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
11 22 33 44
11 22 33 44
11 22 33 44
11 22 33 44
```

12. Task 6k:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
11 11 11 11
22 22 22 22
33 33 33 33
44 44 44 44
```

13. Task 6m: What should be Position 1 and 2 in order to obtain the following output:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
44 44 44 44
33 33 33 33
22 22 22 22
11 11 11 11
```

Position2: dim3 dimGrid(N/4,1,1) ; dim3 dimBlock(N/4 , 1 , 1);

Position1: int index = threadIdx.x + blockIdx.x * blockDim.x;

array[index] = (4- blockIdx.x) * 11;

14. Task 6n:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
11 11 22 22
11 11 22 22
33 33 44 44
33 33 44 44
```

15. Task 6o: What should be Position 1 and 2 in order to obtain the following output:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
22 22 44 44
22 22 44 44
11 11 33 33
11 11 33 33
```

Position2: dim3 dimGrid(N / 8, N / 8, 1); dim3 dimBlock(N / 8, N / 8, 1);

Position1: int index_x = blockIdx.x * blockDim.x + threadIdx.x;

int index_y = blockIdx.y * blockDim.y + threadIdx.y;

array[index_y * blockDim.x * gridDim.x + index_x] = 11 * (blockIdx.x * gridDim.y + (gridDim.y - blockIdx.y - 1) + 1);

Task 7: Matrix Addition

pdc-p200750@lmar:~/cuda

File Actions Edit View Help

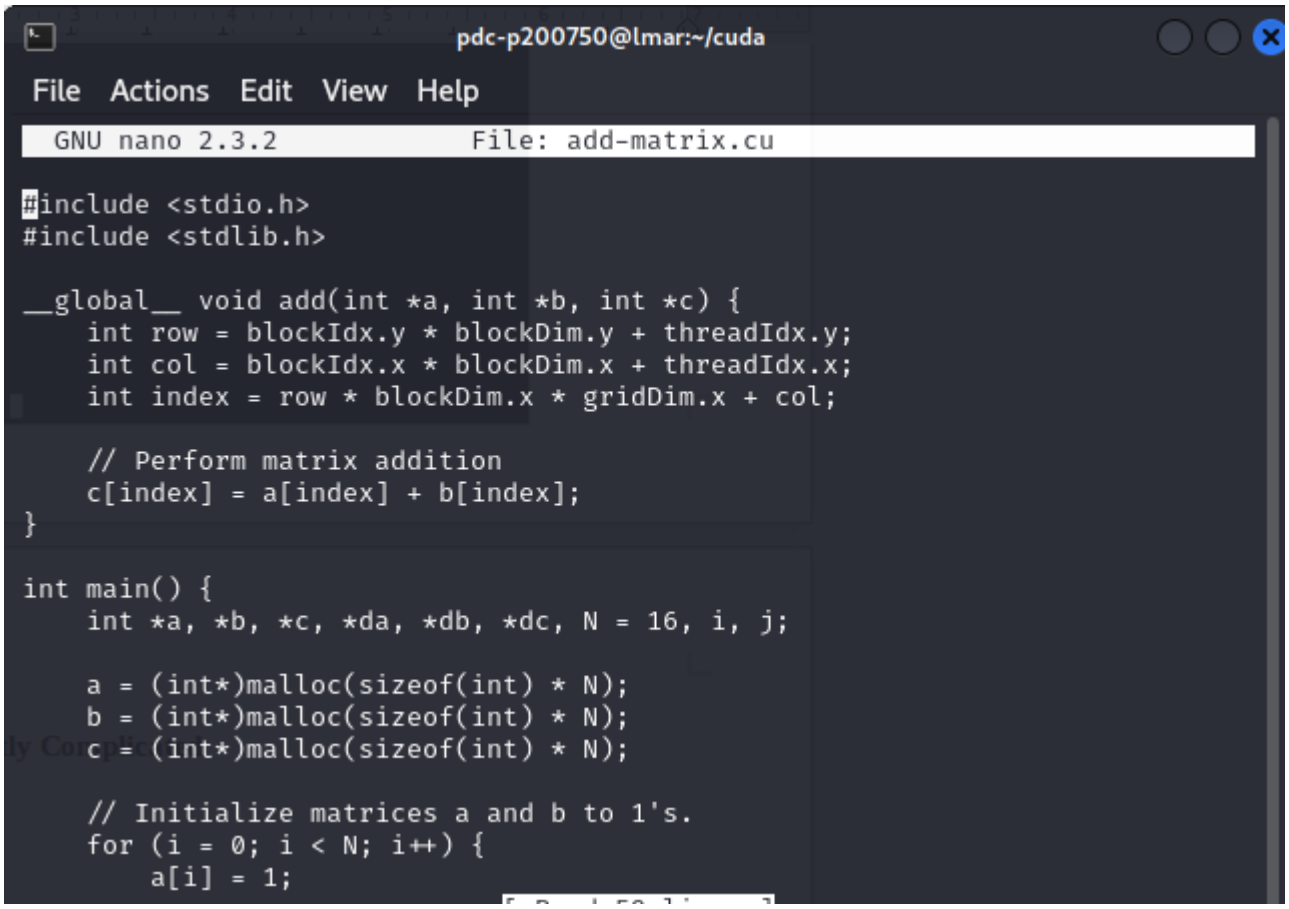
GNU nano 2.3.2 File: matrix-add.cu

```
#include <stdio.h>
#include <stdlib.h>

__global__ void add(int *a, int *b, int *c) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;

    // Perform matrix addition
    c[idx] = a[idx] + b[idx];
}

int main() {
    int *a, *b, *c, *da, *db, *dc, N = 16, i;

    a = (int*)malloc(sizeof(int) * N);
    b = (int*)malloc(sizeof(int) * N);
    c = (int*)malloc(sizeof(int) * N);

    // Initialize matrices a and b to 1's
    for (i = 0; i < N; i++) {
        a[i] = 1;
        b[i] = 1;
    }
```

output:

```
pdc-p200750@lmar ~/cuda $ nvcc matrix-add.cu
pdc-p200750@lmar ~/cuda $ ./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] = 0
a[9] + b[9] = 0
a[10] + b[10] = 0
a[11] + b[11] = 0
a[12] + b[12] = 0
a[13] + b[13] = 0
a[14] + b[14] = 0
a[15] + b[15] = 0
pdc-p200750@lmar ~/cuda $
```

Task 8: Matrix Addition Slightly Complicated



```
GNU nano 2.3.2                File: add-matrix.cu

int main() {
    int *a, *b, *c, *da, *db, *dc, N = 16, i, j;

    a = (int*)malloc(sizeof(int) * N);
    b = (int*)malloc(sizeof(int) * N);
    c = (int*)malloc(sizeof(int) * N);

    // Initialize matrices a and b to 1's.
    for (i = 0; i < N; i++) {
        a[i] = 1;
        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);
```

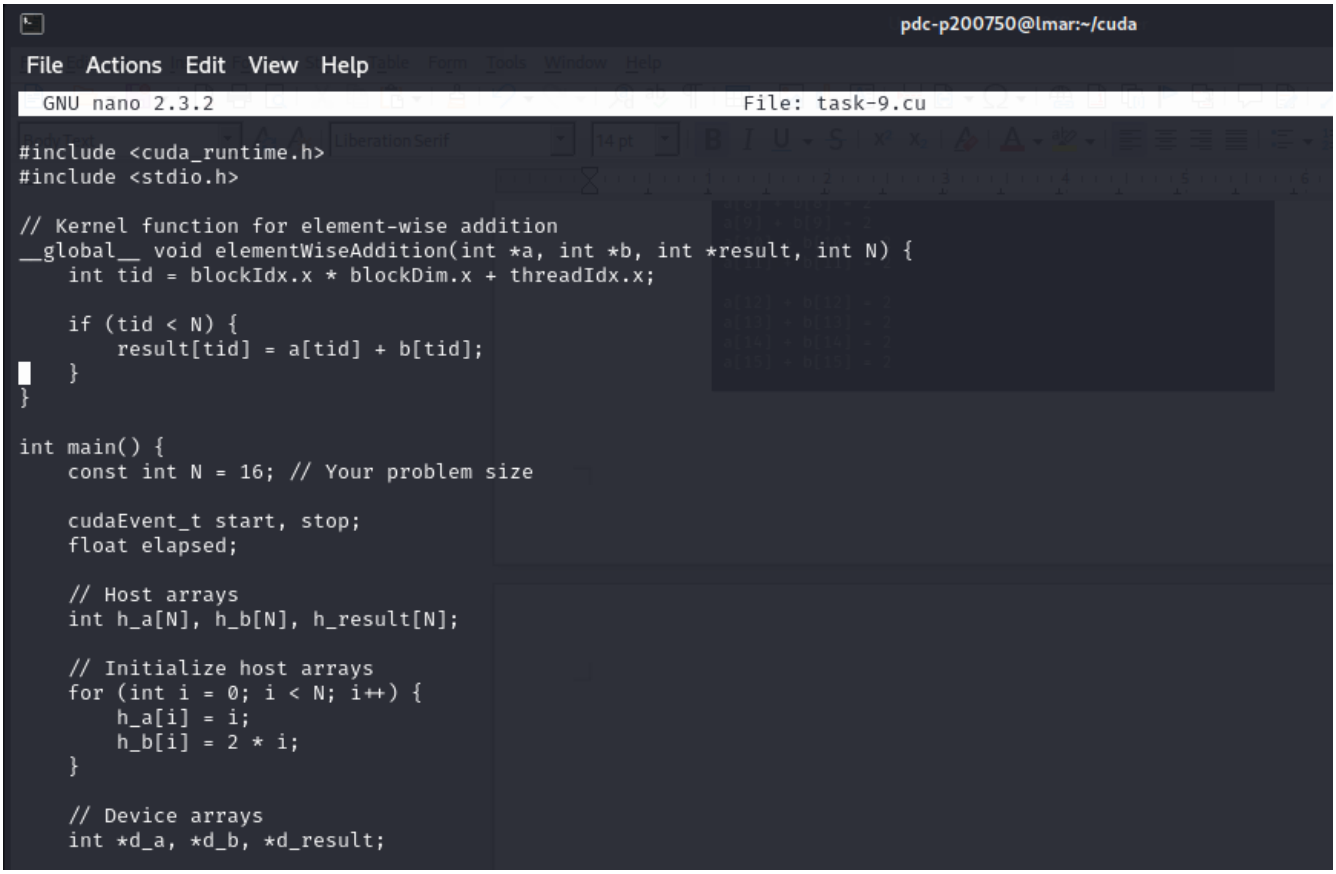
output:

```
pdc-p200750@lmar ~/cuda $ nvcc add-matrix.cu
pdc-p200750@lmar ~/cuda $ ./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
```

Task 9: Measurements

To measure anything in CUDA, we can use the following from the CUDA Events API's:

code:



```
pdc-p200750@lmar:~/cuda
File  Actions  Edit  View  Help
 GNU nano 2.3.2                File: task-9.cu

    cudaMalloc((void **)&d_a, N * sizeof(int));
    cudaMalloc((void **)&d_b, N * sizeof(int));
    cudaMalloc((void **)&d_result, N * sizeof(int));

    // Copy data from host to device
    cudaMemcpy(d_a, h_a, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, h_b, N * sizeof(int), cudaMemcpyHostToDevice);

    printf("Grid X\tGrid Y\tGrid Z\tBlock X\tBlock Y\tBlock Z\tMicroseconds\tMilliseconds\tSeconds\n");

    for (int gridX = 1; gridX <= N; gridX *= 2) {
        for (int gridY = 1; gridY <= N / gridX; gridY *= 2) {
            int gridZ = N / (gridX * gridY);

            for (int blockX = 1; blockX <= N; blockX *= 2) {
                for (int blockY = 1; blockY <= N / blockX; blockY *= 2) {
                    int blockZ = N / (blockX * blockY);

                    dim3 gridSize(gridX, gridY, gridZ);
                    dim3 blockSize(blockX, blockY, blockZ);

                    cudaEventCreate(&start);
                    cudaEventCreate(&stop);

                    cudaEventRecord(start, 0);

                    // Call the Kernel
                    elementWiseAddition<<<gridSize, blockSize>>>(d_a, d_b, d_result, N);
```

output:

```
pdc-p200750@lmar ~/cuda $ nvcc task-9.cu
pdc-p200750@lmar ~/cuda $ ./a.out
Grid X  Grid Y  Grid Z  Block X Block Y Block Z Microseconds    Milliseconds    Seconds
1       1       16      1       1       16      23.94   0.02    0.000024
1       1       16      1       2       8       8.70    0.01    0.000009
1       1       16      1       4       4       6.08    0.01    0.000006
1       1       16      1       8       2       5.79    0.01    0.000006
1       1       16      1       16      1       5.79    0.01    0.000006
1       1       16      2       1       8       5.79    0.01    0.000006
1       1       16      2       2       4       5.95    0.01    0.000006
1       1       16      2       4       2       5.92    0.01    0.000006
1       1       16      2       8       1       5.76    0.01    0.000006
1       1       16      4       1       4       5.76    0.01    0.000006
1       1       16      4       2       2       5.73    0.01    0.000006
1       1       16      4       4       1       5.76    0.01    0.000006
1       1       16      8       1       2       5.98    0.01    0.000006
1       1       16      8       2       1       5.73    0.01    0.000006
1       1       16      16      1       1       5.76    0.01    0.000006
1       2       8       1       1       16      5.79    0.01    0.000006
1       2       8       1       2       8       5.76    0.01    0.000006
1       2       8       1       4       4       5.73    0.01    0.000006
1       2       8       1       8       2       5.79    0.01    0.000006
1       2       8       1       16      1       5.73    0.01    0.000006
1       2       8       2       1       8       5.76    0.01    0.000006
1       2       8       2       2       4       5.95    0.01    0.000006
1       2       8       2       4       2       5.76    0.01    0.000006
1       2       8       2       8       1       5.76    0.01    0.000006
1       2       8       4       1       4       5.79    0.01    0.000006
1       2       8       4       2       2       5.76    0.01    0.000006
1       2       8       4       4       1       5.73    0.01    0.000006
1       2       8       8       1       2       5.86    0.01    0.000006
```