PPL LAB-5 EXERCISES - CUDA

Date @February 13, 2026

NAME : DEVADATHAN N R

ROLL NO : 04

SECTION : CSE A1

REG NO : 230905010

Question 1:

Lab Exercise 1 - Write a CUDA program to add two vectors of length N using:

- N blocks
- N threads

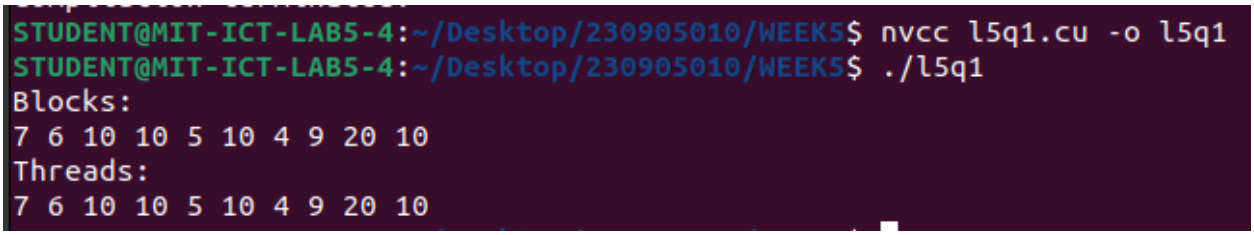
Code:

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include <stdio.h>

const int N = 10;

__global__ void k1(int *a, int *b, int *c)
{
    int id = blockIdx.x * blockDim.x + threadIdx.x;
    if (id < N)
        c[id] = a[id] + b[id];
}

__global__ void k2(int *a, int *b, int *c)
{
    int id = threadIdx.x;
    if (id < N)
```

```
        c[id] = a[id] + b[id];
}

int main()
{
    int *d1, *d2, *d3;
    int s = sizeof(int);

    cudaMalloc((void**)&d1, N * s);
    cudaMalloc((void**)&d2, N * s);
    cudaMalloc((void**)&d3, N * s);

    int h1[N] = {5, -3, 12, 7, 0, 14, -6, 8, 21, 4};
    int h2[N] = {2, 9, -2, 3, 5, -4, 10, 1, -1, 6};
    int h3[N];

    cudaMemcpy(d1, h1, N * s, cudaMemcpyHostToDevice);
    cudaMemcpy(d2, h2, N * s, cudaMemcpyHostToDevice);

    k1<<<N,1>>>(d1, d2, d3);
    cudaMemcpy(h3, d3, N * s, cudaMemcpyDeviceToHost);

    printf("Blocks:\n");
    for(int i = 0; i < N; i++)
        printf("%d ", h3[i]);
    printf("\n");

    k2<<<1,N>>>(d1, d2, d3);
    cudaMemcpy(h3, d3, N * s, cudaMemcpyDeviceToHost);

    printf("Threads:\n");
    for(int i = 0; i < N; i++)
        printf("%d ", h3[i]);
    printf("\n");

    cudaFree(d1);
```

```
    cudaFree(d2);
    cudaFree(d3);

    return 0;
}
```

Output:



Question 2:

Lab Exercise 2 - Implement a CUDA program to add two vectors of length N keeping number of threads per block as 256 (constant) and varying number of blocks to handle N elements.

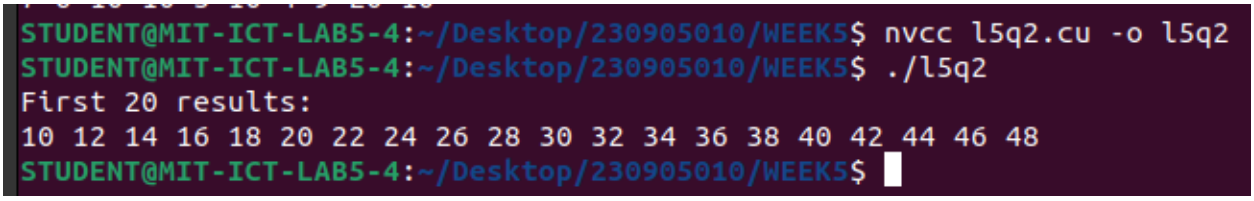
Code:

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include <stdio.h>

const int N = 1000;

__global__ void k(int *a, int *b, int *c)
{
    int id = blockIdx.x * blockDim.x + threadIdx.x;
    if (id < N)
        c[id] = a[id] + b[id];
}
```

```
int main()
{
    int *d1, *d2, *d3;
    int s = sizeof(int);

    cudaMalloc((void**)&d1, N * s);
    cudaMalloc((void**)&d2, N * s);
    cudaMalloc((void**)&d3, N * s);

    int h1[N], h2[N], h3[N];

    for (int i = 0; i < N; i++)
    {
        h1[i] = i;
        h2[i] = i + 10;
    }

    cudaMemcpy(d1, h1, N * s, cudaMemcpyHostToDevice);
    cudaMemcpy(d2, h2, N * s, cudaMemcpyHostToDevice);

    int t = 256;
    int b = (N + t - 1) / t;

    k<<<b, t>>>(d1, d2, d3);

    cudaMemcpy(h3, d3, N * s, cudaMemcpyDeviceToHost);

    printf("First 20 results:\n");
    for (int i = 0; i < 20; i++)
        printf("%d ", h3[i]);
    printf("\n");

    cudaFree(d1);
    cudaFree(d2);
    cudaFree(d3);
```

```
    return 0;
}
```

Output:



Question 3:

Lab Exercise 3 - Write a CUDA program to process a 1D array containing angles in radians and generate sine values in the output array.

Code:

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include <stdio.h>
#include <math.h>

#define PI 3.14159265359

const int N = 6;

__global__ void sine_calc(double *x, double *y){
    int i = threadIdx.x;
    if(i < N){
        y[i] = sin(x[i]);
    }
}

int main(){
    double *d_x, *d_y;
```

```
    int sz = sizeof(double);

    cudaMalloc((void**)&d_x, N * sz);
    cudaMalloc((void**)&d_y, N * sz);

    double h_x[N] = {PI/8, PI/5, PI/2, 3*PI/4, PI, 2*PI};
    double h_y[N];

    cudaMemcpy(d_x, h_x, N * sz, cudaMemcpyHostToDevice);

    sine_calc<<<1,N>>>(d_x, d_y);

    cudaMemcpy(h_y, d_y, N * sz, cudaMemcpyDeviceToHost);

    printf("Sine Results:\n");
    for(int i = 0; i < N; i++){
        printf("sin(%.4lf) = %.4lf\n", h_x[i], h_y[i]);
    }

    cudaFree(d_x);
    cudaFree(d_y);

    return 0;
}
```

Output:

```
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK5$ nvcc l5q3.cu -o l5q3
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK5$ ./l5q3
Sine Results:
sin(0.3927) = 0.3827
sin(0.6283) = 0.5878
sin(1.5708) = 1.0000
sin(2.3562) = 0.7071
sin(3.1416) = -0.0000
sin(6.2832) = 0.0000
```