Name : Tushar Pathak
RegNumber : 230905396
Roll Number: B48

Lab 5: Programs on arrays in CUDA

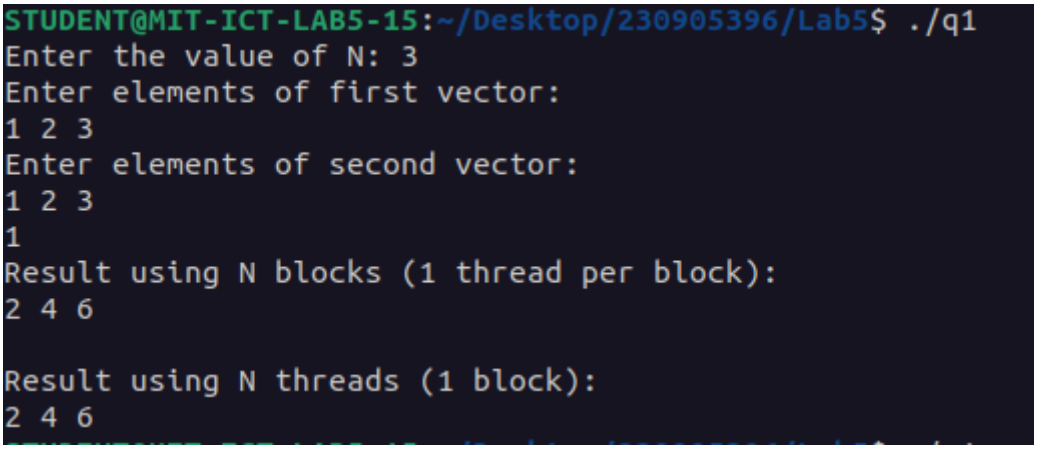
Q1)Write a program in CUDA to add two vectors of length N using
a) block size as N
b) N threads

Code:

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

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

__global__ void vector_additionNthreads(int *a, int *b, int *c, int n){
    int tid = threadIdx.x;
    if(tid < n)
        c[tid] = a[tid] + b[tid];
}

int main(){
    int n;

    printf("Enter the value of N: ");
    scanf("%d", &n);

    int size = n * sizeof(int);

    int *a = (int*)malloc(size);
    int *b = (int*)malloc(size);
    int *c = (int*)malloc(size);

    printf("Enter elements of first vector:\n");
    for(int i = 0; i < n; i++){
        scanf("%d", &a[i]);
    }

    printf("Enter elements of second vector:\n");
      for(int i = 0; i < n; i++){
        scanf("%d", &b[i]);
    }
```

```
    int *d_a, *d_b, *d_c;

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

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

    vector_additionNblocks<<<n, 1>>>(d_a, d_b, d_c, n);
    cudaMemcpy(c, d_c, size, cudaMemcpyDeviceToHost);

    printf("\nResult using N blocks (1 thread per block):\n");
    for(int i = 0; i < n; i++){
        printf("%d ", c[i]);
    }
    printf("\n");

    vector_additionNthreads<<<1, n>>>(d_a, d_b, d_c, n);
    cudaMemcpy(c, d_c, size, cudaMemcpyDeviceToHost);

    printf("\nResult using N threads (1 block):\n");
    for(int i = 0; i < n; i++){
        printf("%d ", c[i]);
    }
    printf("\n");

    cudaFree(d_a);
    cudaFree(d_b);
    cudaFree(d_c);

    free(a);
    free(b);
    free(c);

    return 0;
}
```

Output :



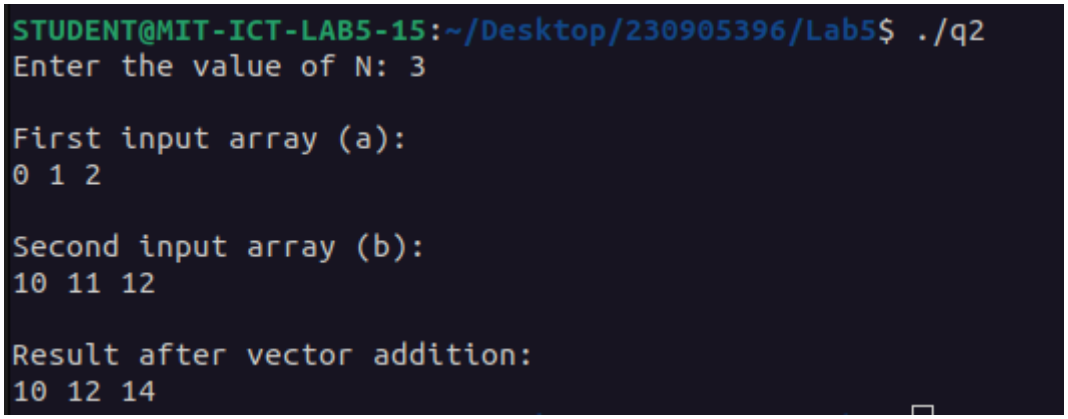
Q2) Implement a CUDA program to add two vectors of length N by keeping the number of threads per block as 256 (constant) and vary the number of blocks to handle N elements.

Code:

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

#define THREADS_PER_BLOCK 256

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

int main(){
    int n;

    printf("Enter the value of N: ");
    scanf("%d", &n);

    int size = n * sizeof(int);

    int *a = (int*)malloc(size);
    int *b = (int*)malloc(size);
    int *c = (int*)malloc(size);

    for(int i = 0; i < n; i++){
        a[i] = i;
        b[i] = i + 10;
    }

    int *d_a, *d_b, *d_c;

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

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

    int blocks = ceil((float)n / THREADS_PER_BLOCK);

    vector_addition<<<blocks, THREADS_PER_BLOCK>>>(d_a, d_b, d_c, n);

    cudaMemcpy(c, d_c, size, cudaMemcpyDeviceToHost);

    printf("\nFirst input array (a):\n");
```

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

    printf("\nSecond input array (b):\n");
    for(int i = 0; i < n; i++){
        printf("%d ", b[i]);
    }
    printf("\n");

    printf("\nResult after vector addition:\n");
    for(int i = 0; i < n; i++){
        printf("%d ", c[i]);
    }
    printf("\n");

    cudaFree(d_a);
    cudaFree(d_b);
    cudaFree(d_c);

    free(a);
    free(b);
    free(c);

    return 0;
}
```

Output:

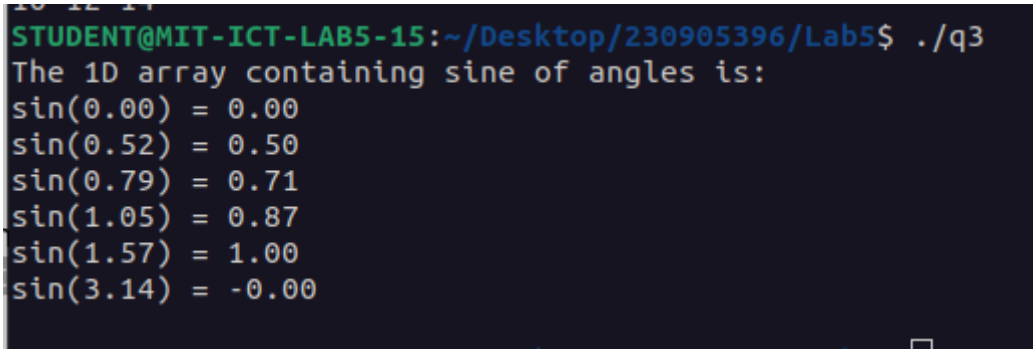


Q3)Write a program in CUDA to process a ID array containing angles in radians to generate sine of the angles in the output array. Use appropriate function.

Code:

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include<stdio.h>
#define PI 3.1415

const int n = 6;

__global__ void angles_to_sine(double *a, double *b){
    int tid = threadIdx.x;
    if(tid < n) {
        b[tid] = sinf(a[tid]);
    }
}

int main(){
    double *d_a, *d_b;
    int size = sizeof(double);

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

    double a[n] = {0, PI/6, PI/4, PI/3, PI/2, PI};
    double b[n];

    cudaMemcpy(d_a, a, n * size, cudaMemcpyHostToDevice);

    angles_to_sine<<<1,n>>>(d_a, d_b);

    cudaMemcpy(b, d_b, n * size, cudaMemcpyDeviceToHost);

    printf("The 1D array containing sine of angles is: \n");
    for(int i = 0; i < n; i++){
        printf("sin(%.2lf) = %.2lf \n",a[i],b[i]);
    }
    printf("\n");

    cudaFree(d_a);
    cudaFree(d_b);
    return 0;
}
```

Output:


STUDENT@MIT-ICT-LAB5-15:~/Desktop/230905396/Lab5$ ./q3
The 1D array containing sine of angles is:
sin(0.00) = 0.00
sin(0.52) = 0.50
sin(0.79) = 0.71
sin(1.05) = 0.87
sin(1.57) = 1.00
sin(3.14) = -0.00