Class: B.E. A
Batch: A1
Name: Aryan Ghatge
Roll No.: 4101005
LP-V HPC lab-1

********** CODE **********

```
#include <iostream>
#include <cuda_runtime.h>
using namespace std;

__global__ void vectorAdd(int *A, int *B, int *C, int N) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < N) C[i] = A[i] + B[i];
}

int main() {
    int N;
    cout << "Enter the size of vectors: "; cin >> N;
    int *h_A = new int[N], *h_B = new int[N], *h_C = new int[N];

    cout << "Enter elements of vector A:\n";
    for (int i = 0; i < N; i++) cin >> h_A[i];

    cout << "Enter elements of vector B:\n";
    for (int i = 0; i < N; i++) cin >> h_B[i];

    int *d_A, *d_B, *d_C;
    cudaMalloc(&d_A, N * sizeof(int));
    cudaMalloc(&d_B, N * sizeof(int));
    cudaMalloc(&d_C, N * sizeof(int));

    cudaMemcpy(d_A, h_A, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_B, h_B, N * sizeof(int), cudaMemcpyHostToDevice);

    vectorAdd<<<(N + 255) / 256, 256>>>(d_A, d_B, d_C, N);
    cudaMemcpy(h_C, d_C, N * sizeof(int), cudaMemcpyDeviceToHost);

    cout << "Result of vector addition:\n";
    for (int i = 0; i < N; i++) cout << h_C[i] << " ";
    cout << endl;
```

```
    cudaFree(d_A); cudaFree(d_B); cudaFree(d_C);
    delete[] h_A; delete[] h_B; delete[] h_C;

    return 0;
}
```

********** **OUTPUT** **********

```
Enter the size of vectors: 5
Enter elements of vector A:
1 2 3 4 5
Enter elements of vector B:
10 20 30 40 50

Result of vector addition:
11 22 33 44 55
```