#include <iostream>

#include <vector>

#include <cuda\_runtime.h>

\_\_global\_\_ void vectorAddition(const int\* a, const int\* b, int\* c, int size) {

int tid = blockIdx.x \* blockDim.x + threadIdx.x;

if (tid < size) {

c[tid] = a[tid] + b[tid]; }}

int main() {

int size = 1000000;

int blockSize = 256;

std::vector<int> hostA(size, 1);

std::vector<int> hostB(size, 2);

std::vector<int> hostC(size, 0);

int\* deviceA, \* deviceB, \* deviceC;

cudaMalloc((void\*\*)&deviceA, size \* sizeof(int));

cudaMalloc((void\*\*)&deviceB, size \* sizeof(int));

cudaMalloc((void\*\*)&deviceC, size \* sizeof(int));

cudaMemcpy(deviceA, hostA.data(), size \* sizeof(int), cudaMemcpyHostToDevice);

cudaMemcpy(deviceB, hostB.data(), size \* sizeof(int), cudaMemcpyHostToDevice);

vectorAddition<<<(size + blockSize - 1) / blockSize, blockSize>>>(deviceA, deviceB, deviceC, size);

cudaMemcpy(hostC.data(), deviceC, size \* sizeof(int), cudaMemcpyDeviceToHost);

for (int i = 0; i < size; i++) {

std::cout << hostC[i] << " ";}

std::cout << std::endl;

cudaFree(deviceA);

cudaFree(deviceB);

cudaFree(deviceC);

return 0;}