~\Desktop\Outputs\HPC\4-A.cpp

Name:Arshin Mokashi
Roll No.:COBB26

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

__global__ void addVectors(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 = 1000000;
    int* A, * B, * C;
    int size = n * sizeof(int);

    cudaMallocHost(&A, size);
    cudaMallocHost(&B, size);
    cudaMallocHost(&C, size);

    for (int i = 0; i < n; i++) {
        A[i] = i;
        B[i] = i * 2;
    }

    int* dev_A, * dev_B, * dev_C;
    cudaMalloc(&dev_A, size);
    cudaMalloc(&dev_B, size);
    cudaMalloc(&dev_C, size);

    cudaMemcpy(dev_A, A, size, cudaMemcpyHostToDevice);
    cudaMemcpy(dev_B, B, size, cudaMemcpyHostToDevice);

    int blockSize = 256;
    int numBlocks = (n + blockSize - 1) / blockSize;
    addVectors<<<numBlocks, blockSize>>>(dev_A, dev_B, dev_C, n);

    cudaDeviceSynchronize();

    cudaMemcpy(C, dev_C, size, cudaMemcpyDeviceToHost);

    for (int i = 0; i < 10; i++) {
        cout << C[i] << " ";
    }
    cout << endl;

    cudaFree(dev_A);
    cudaFree(dev_B);
    cudaFree(dev_C);
    cudaFreeHost(A);
    cudaFreeHost(B);
    cudaFreeHost(C);

    return 0;
}

/*

Output:

0 3 6 9 12 15 18 21 24 27

*/
```