Practical-04 Matrix Multiplication

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <cassert>
#include "cuda_runtime.h"
#include "device_launch_parameters.h"


#define BLOCK_SIZE 16

__global__ void matrixMultiplication(int* a, int* b, int* c, int N) {
    int row = blockIdx.y * blockDim.y + threadIdx.y;
    int col = blockIdx.x * blockDim.x + threadIdx.x;

    if (row < N && col < N) {
        int sum = 0;
        for (int i = 0; i < N; ++i) {
            sum += a[row * N + i] * b[i * N + col];
        }
        c[row * N + col] = sum;
    }
}

void verifyResult(std::vector<int>& a, std::vector<int>& b, std::vector<int>& c, int
N) {
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            int sum = 0;
            for (int k = 0; k < N; ++k) {
                sum += a[i * N + k] * b[k * N + j];
            }
            assert(c[i * N + j] == sum);
        }
    }
}

int main() {
    const int N = 4;
    const int size = N * N;

    std::vector<int> h_A(size);
    std::vector<int> h_B(size);
    std::vector<int> h_C(size);


    srand(static_cast<unsigned>(time(NULL)));
    for (int i = 0; i < size; ++i) {
        h_A[i] = rand() % 10;
        h_B[i] = rand() % 10;
    }

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

```
    cudaMemcpy(d_A, h_A.data(), size * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_B, h_B.data(), size * sizeof(int), cudaMemcpyHostToDevice);


    dim3 blockSize(BLOCK_SIZE, BLOCK_SIZE);
    dim3 gridSize((N + BLOCK_SIZE - 1) / BLOCK_SIZE, (N + BLOCK_SIZE - 1) /
BLOCK_SIZE);

    matrixMultiplication << <gridSize, blockSize >> > (d_A, d_B, d_C, N);

    cudaMemcpy(h_C.data(), d_C, size * sizeof(int), cudaMemcpyDeviceToHost);

    verifyResult(h_A, h_B, h_C, N);

    std::cout << "Result matrix (some elements):\n";
    for (int i = 0; i < std::min(N, 4); ++i) {
        for (int j = 0; j < std::min(N, 4); ++j) {
            std::cout << h_C[i * N + j] << " ";
        }
        std::cout << "\n";
    }
    cudaFree(d_A);
    cudaFree(d_B);
    cudaFree(d_C);

    std::cout << "Matrix multiplication completed successfully.\n";

    return 0;
}
```

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

Result matrix (some elements):

107 144 91 76

61 142 42 42

80 158 61 56

135 153 136 108

Matrix multiplication completed successfully.

C:\Users\DELL\source\repos\CudaRuntime2\x64\Debug\CudaRuntime2.exe (process 15296) exited with code 0.

Press any key to close this window . . .