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 matrixMul(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 k = 0; k < n; k++)
            sum += A[row * n + k] * B[k * n + col];
        C[row * n + col] = sum;
    }
}

int main() {
    int N;
    cout << "Enter the size of the square matrix: "; cin >> N;

    int *h_A = new int[N * N], *h_B = new int[N * N], *h_C = new int[N * N];

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

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

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

    cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
```

```
    cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);

    dim3 threadsPerBlock(16, 16);
    dim3 blocksPerGrid((N + 15) / 16, (N + 15) / 16);

    matrixMul<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, N);
    cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);

    cout << "Result of matrix multiplication:\n";
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) cout << h_C[i * N + j] << " ";
        cout << "\n";
    }

    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 the square matrix: 3
Enter elements of matrix A:
1 2 3
4 5 6
7 8 9
Enter elements of matrix B:
9 8 7
6 5 4
3 2 1

Result of matrix multiplication:
30 24 18
84 69 54
138 114 90
```