```
//Write a cuda program for
//2. Matrics Multiplication using CUDA C

#include <iostream>
#include <cuda.h>

#define N 1024

__global__ void multiplyMatrices(int *A, int *B, int *C, int N) {
    int row = threadIdx.y + blockIdx.y * blockDim.y;
    int col = threadIdx.x + blockIdx.x * blockDim.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 *A, *B, *C, *d_A, *d_B, *d_C;
    A = new int[N*N]; B = new int[N*N]; C = new int[N*N];

    for (int i = 0; i < N * N; i++) { A[i] = 1; B[i] = 1; }

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

    cudaMemcpy(d_A, A, N * N * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_B, B, N * N * sizeof(int), cudaMemcpyHostToDevice);

    dim3 blockSize(16, 16);
    dim3 gridSize((N + blockSize.x - 1) / blockSize.x, (N + blockSize.y - 1) /
blockSize.y);
    multiplyMatrices<<<gridSize, blockSize>>>(d_A, d_B, d_C, N);

    cudaMemcpy(C, d_C, N * N * sizeof(int), cudaMemcpyDeviceToHost);

    std::cout << "C[0] = " << C[0] << ", C[1] = " << C[1] << std::endl;  //
Example output
    delete[] A; delete[] B; delete[] C;
    cudaFree(d_A); cudaFree(d_B); cudaFree(d_C);
    return 0;
}
```