```
#Name: Saloni Satappa Bailkar
#Div: A Roll No. COBA013
#HPC LAB Assignment No. 04(b)

!nvcc --version
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2023 NVIDIA Corporation
Built on Tue_Aug_15_22:02:13_PDT_2023
Cuda compilation tools, release 12.2, V12.2.140
Build cuda_12.2.r12.2/compiler.33191640_0
```

```
# Define the CUDA code as a string
cuda_code = """
#include <iostream>
#include <cuda_runtime.h>
using namespace std;


__global__ void matmul(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 Pvalue = 0;
        for (int k = 0; k < N; k++) {
            Pvalue += A[Row * N + k] * B[k * N + Col];
        }
        C[Row * N + Col] = Pvalue;
    }
}

int main() {
    int N = 10;
    int size = N * N * sizeof(int);
    int* A, * B, * C;
    int* dev_A, * dev_B, * dev_C;
    cudaMallocHost(&A, size);
    cudaMallocHost(&B, size);
    cudaMallocHost(&C, size);
    cudaMalloc(&dev_A, size);
    cudaMalloc(&dev_B, size);
    cudaMalloc(&dev_C, size);
    // Initialize matrices A and B
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            A[i * N + j] = i * N + j;
            B[i * N + j] = j * N + i;
        }
    }
    cudaMemcpy(dev_A, A, size, cudaMemcpyHostToDevice);
    cudaMemcpy(dev_B, B, size, cudaMemcpyHostToDevice);
    dim3 dimBlock(16, 16);
    dim3 dimGrid((N + dimBlock.x - 1) / dimBlock.x, (N + dimBlock.y - 1) / dimBlock.y);
    matmul<<<dimGrid, dimBlock>>>(dev_A, dev_B, dev_C, N);
    cudaDeviceSynchronize(); // Ensure kernel execution is complete
    cudaMemcpy(C, dev_C, size, cudaMemcpyDeviceToHost);
   // Print the result
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        std::cout << C[i * N + j] << " ";
    }
    std::cout << std::endl;
}


    // Free memory
    cudaFree(dev_A);
    cudaFree(dev_B);
    cudaFree(dev_C);
    cudaFreeHost(A);
    cudaFreeHost(B);
    cudaFreeHost(C);
    return 0;
}
"""
```

```
# Write the CUDA code to a file
with open('cuda_code.cu', 'w') as f:
    f.write(cuda_code)

# Compile the CUDA code
!nvcc -o cuda_program cuda_code.cu

# Execute the compiled program
!./cuda_program
```

```
285 735 1185 1635 2085 2535 2985 3435 3885 4335
735 2185 3635 5085 6535 7985 9435 10885 12335 13785
1185 3635 6085 8535 10985 13435 15885 18335 20785 23235
1635 5085 8535 11985 15435 18885 22335 25785 29235 32685
2085 6535 10985 15435 19885 24335 28785 33235 37685 42135
2535 7985 13435 18885 24335 29785 35235 40685 46135 51585
2985 9435 15885 22335 28785 35235 41685 48135 54585 61035
3435 10885 18335 25785 33235 40685 48135 55585 63035 70485
3885 12335 20785 29235 37685 46135 54585 63035 71485 79935
4335 13785 23235 32685 42135 51585 61035 70485 79935 89385
```

Start coding or generate with AI.