Name	**Huzaifa Ahmad**
Registration No	**FA20-BCS-081**
Submitted To	**Ezhaz Mustafa**
Subject	**Parallel and Distributed Computing**

Question

1. **Matrix Multiplication:** Implement a CUDA program for multiplication on two large matrices.

Steps and Requirements:

1. **Matrix Multiplication (60% of the grade):**
   - Write a CUDA program to multiply two square matrices AAA and BBB of size N×NN \times NN×N.
   - Optimize your program to handle large matrices (e.g., N=1024N = 1024N=1024).
   - Compare the performance of your GPU implementation with a sequential CPU implementation.
   - Measure and report the execution time for both implementations.
   - Explain the observed performance difference and the impact of GPU architecture on matrix multiplication.

Code

```
import numpy as np
import cupy as cp
import time

# Define the matrix size
N = 1024

# Generate two random matrices
A_cpu = np.random.rand(N, N).astype(np.float32)
B_cpu = np.random.rand(N, N).astype(np.float32)

# Measure the time for CPU matrix multiplication
start_cpu = time.time()
C_cpu = np.dot(A_cpu, B_cpu)
end_cpu = time.time()
cpu_time = end_cpu - start_cpu

print(f"CPU time: {cpu_time:.4f} seconds")

# Transfer the matrices to the GPU
A_gpu = cp.array(A_cpu)
B_gpu = cp.array(B_cpu)

# Measure the time for GPU matrix multiplication
```

```
start_gpu = cp.cuda.Event()
end_gpu = cp.cuda.Event()

start_gpu.record()
C_gpu = cp.dot(A_gpu, B_gpu)
end_gpu.record()

end_gpu.synchronize()
gpu_time = cp.cuda.get_elapsed_time(start_gpu, end_gpu) / 1000  # Convert to seconds

print(f"GPU time: {gpu_time:.4f} seconds")

# Verify the results are the same
C_cpu_from_gpu = cp.asnumpy(C_gpu)
assert np.allclose(C_cpu, C_cpu_from_gpu), "Matrices are not equal!"

print("Results are the same!")
```