Assignment 4 B

```
#include &lt;cuda_runtime.h&gt;
#include &lt;iostream&gt;
__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 &lt; N &amp;&amp; Col &lt; N) {
int Pvalue = 0;
for (int k = 0; k &lt; N; k++) {
Pvalue += A[Row*N+k] * B[k*N+Col];
}
C[Row*N+Col] = Pvalue;
}
}
int main() {
int N = 512;
int size = N * N * sizeof(int);
int* A, * B, * C;
int* dev_A, * dev_B, * dev_C;
cudaMallocHost(&amp;A, size);
cudaMallocHost(&amp;B, size);
cudaMallocHost(&amp;C, size);
cudaMalloc(&amp;dev_A, size);
cudaMalloc(&amp;dev_B, size);
cudaMalloc(&amp;dev_C, size);
// Initialize matrices A and B
for (int i = 0; i &lt; N; i++) {
for (int j = 0; j &lt; 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, N/dimBlock.y);
matmul&lt;&lt;&lt;dimGrid, dimBlock&gt;&gt;&gt;(dev_A, dev_B,
dev_C, N);
cudaMemcpy(C, dev_C
// Print the result
for (int i = 0; i &lt; 10; i++) {
for (int j = 0; j &lt; 10; j++) {
std::cout &lt;&lt; C[i*N+j] &lt;&lt; &quot; &quot;;
}
std::cout &lt;&lt; std::endl;
}
// Free memory
cudaFree(dev_A);
cudaFree(dev_B);

cudaFree(dev_C);
cudaFreeHost(A);
cudaFreeHost(B);
cudaFreeHost(C);
return 0;
```