```
# Set up CUDA
#First Change runtime to GPU and run this cell
!pip install git+https://github.com/afnan47/cuda.git
%load_ext nvcc_plugin
```

```
Collecting git+https://github.com/afnan47/cuda.git
  Cloning https://github.com/afnan47/cuda.git to /tmp/pip-req-build-ne3ms1ap
  Running command git clone --filter=blob:none --quiet https://github.com/afnan47/cuda.git /tmp/pip-req-build-ne3ms1ap
  Resolved https://github.com/afnan47/cuda.git to commit aac710a35f52bb78ab34d2e52517237941399eff
  Preparing metadata (setup.py) ... done
The nvcc_plugin extension is already loaded. To reload it, use:
  %reload_ext nvcc_plugin
```

```
%%writefile vector_add.cu
#include <iostream>

using namespace std;

__global__
void add(int* A, int* B, int* C, int size) {
    int tid = blockIdx.x * blockDim.x + threadIdx.x;
    if (tid < size) {
        C[tid] = A[tid] + B[tid];
    }
}

void initialize(int* vector, int size) {
    for (int i = 0; i < size; i++) {
        vector[i] = rand() % 10;
    }
}

void print(int* vector, int size) {
    for (int i = 0; i < size; i++) {
        cout << vector[i] << " ";
    }
    cout << endl;
}

 int main() {
    int N = 4;
    int* A = new int[N];
    int* B = new int[N];
    int* C = new int[N];

    initialize(A, N);
    initialize(B, N);

    cout << "Vector A: ";
    print(A, N);
    cout << "Vector B: ";
    print(B, N);

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


    cudaMemcpy(d_A, A, bytes, cudaMemcpyHostToDevice);
    cudaMemcpy(d_B, B, bytes, cudaMemcpyHostToDevice);

    int threadsPerBlock = 256;
    int blocksPerGrid = (N + threadsPerBlock - 1) / threadsPerBlock;
    add<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, N);
    cudaDeviceSynchronize();

    cudaMemcpy(C, d_C, bytes, cudaMemcpyDeviceToHost);

    cudaError_t err = cudaGetLastError();
    if (err != cudaSuccess) {
        cout << "CUDA Error: " << cudaGetErrorString(err) << endl;
        return 1;
    }

    cout << "Addition: ";
    print(C, N);

    delete[] A;
    delete[] B;
```

```
    delete[] C;
    cudaFree(d_A);
    cudaFree(d_B);
    cudaFree(d_C);

    return 0;
}
```

```
Overwriting vector_add.cu
```

```
!nvcc -arch=sm_75 vector_add.cu -o vector_add
!./vector_add
```

```
Vector A: 3 6 7 5
Vector B: 3 5 6 2
Addition: 6 11 13 7
```