```
!pip install git+https://github.com/afnan47/cuda.git
```

```
  Collecting git+https://github.com/afnan47/cuda.git
    Cloning https://github.com/afnan47/cuda.git to /tmp/pip-req-build-xtepx2nb
    Running command git clone --filter=blob:none --quiet https://github.com/afnan47/cuda.git /tmp/pip-req-build-xtepx2nb
    Resolved https://github.com/afnan47/cuda.git to commit aac710a35f52bb78ab34d2e52517237941399eff
    Preparing metadata (setup.py) ... done
  Building wheels for collected packages: NVCCPlugin
    Building wheel for NVCCPlugin (setup.py) ... done
    Created wheel for NVCCPlugin: filename=NVCCPlugin-0.0.2-py3-none-any.whl size=4289 sha256=c7065fdd288f2d53cedfead4cd6439d650b326ce6f6e53c1c20d2d
    Stored in directory: /tmp/pip-ephem-wheel-cache-e28zagq_/wheels/aa/f3/44/e10c1d226ec561d971fcd4b0463f6bff08602afa928a3e7bc7
  Successfully built NVCCPlugin
  Installing collected packages: NVCCPlugin
  Successfully installed NVCCPlugin-0.0.2
```

```
%load_ext nvcc_plugin
```

```
  created output directory at /content/src
  Out bin /content/result.out
```

```
%%cu

#include <bits/stdc++.h>
#include <iostream>

using namespace std;

__global__ void vectorAdd(int *a, int*b, int *results, int n)
{
    int tid = threadIdx.x + blockIdx.x * blockDim.x;
    if(tid < n)
    {
        results[tid] = a[tid] + b[tid];
    }
}

void printArr(int *arr, int n)
{
    for(int i=0; i<n; i++)
    {
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}

int main()
{
    int *a, *b, *c;
    int *a_dev, *b_dev, *c_dev;
    int n = 10;

    // Initialize host(CPU) memory
    a = new int[n];
    b = new int[n];
    c = new int[n];

    // Initialize device(GPU) memory
    int size = n * sizeof(int);
    cudaMalloc(&a_dev, size);
    cudaMalloc(&b_dev, size);
    cudaMalloc(&c_dev, size);

    // Adding Data in host memory
    for(int i=0; i<n; ++i)
    {
        a[i] = 1;
        b[i] = 2;
    }

    // Transfer input data from host(CPU) to device(GPU)
    cudaMemcpy(a_dev, a, size, cudaMemcpyHostToDevice);
    cudaMemcpy(b_dev, b, size, cudaMemcpyHostToDevice);

    // Calling vectorAdd()
    int threads = 1024;
    int blocks = ( n - 1 + threads) / threads;
    vectorAdd<<<blocks, threads>>>(a_dev, b_dev, c_dev, n);

    // Transfer results data from device(GPU) to host(CPU)
    cudaMemcpy(c, c_dev, size, cudaMemcpyDeviceToHost);

    cout<<"Array (a): ";
    printArr(a, n);
    cout<<"Array (b): ";
    printArr(b, n);
```

```
    bool status = true;
    cout<<"Array (c): ";
    for(int i=0; i<n; ++i)
    {
        cout<<c[i]<<" ";
        if(c[i] != a[i] + b[i])
        {
            status = false;
            cout<<"Vector Addition Failed";
        }
    }
    cout<<endl;
    if(status = true)
    {
        cout<<"Vector Addition Successfull";
    }

    return 0;
}
```

```
 Array (a): 1 1 1 1 1 1 1 1 1 1
 Array (b): 2 2 2 2 2 2 2 2 2 2
 Array (c): 3 3 3 3 3 3 3 3 3 3
 Vector Addition Successfull
```

Start coding or generate with AI.