

```
#Amruta Sool
#BBC019128
```

```
!nvcc --version
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Sep_21_10:33:58_PDT_2022
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
```

```
!pip install git+https://github.com/andreinechaev/nvcc4jupyter.git
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting git+https://github.com/andreinechaev/nvcc4jupyter.git
  Cloning https://github.com/andreinechaev/nvcc4jupyter.git to /tmp/pip-req-build-777x2ip2
  Running command git clone --filter=blob:none --quiet https://github.com/andreinechaev/nvcc4jupyter.git /tmp/pip-req-build-777x2ip2
  Resolved https://github.com/andreinechaev/nvcc4jupyter.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=4287 sha256=930afb886e8d56092eff3d550413d136fab0d184d8986fbb74c186
  Stored in directory: /tmp/pip-ephem-wheel-cache-7hnkrvum/wheels/a8/b9/18/23f8ef71ceb0f63297dd1903aedd067e6243a68ea756d6f6ea
Successfully built NVCCPlugin
Installing collected packages: NVCCPlugin
Successfully installed NVCCPlugin-0.0.2
```

```
# Commented out IPython magic to ensure Python compatibility.
%load_ext nvcc_plugin
```

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

```
#Commented out IPython magic to ensure Python compatibility.
```

```
%%cu
#include<stdio.h>
#include<cuda.h>
#include<stdlib.h>
#include<time.h>

__global__ void max1(int* input)
{
    const int tid = threadIdx.x;

    auto step_size = 1;
    int number_of_threads = blockDim.x;
    int temp;

    while (number_of_threads > 0)
    {
        if (tid < number_of_threads) // still alive?
        {
            const auto fst = tid * step_size * 2;
            const auto snd = fst + step_size;
            //input[fst] += input[snd];
            if (input[fst]<input[snd])
            {
                temp=input[fst];
                input[fst]=input[snd];
                input[snd]=temp;
            }
        }
        __syncthreads();
        step_size <<= 1;
        number_of_threads >>= 1;
    }
}
```

```
int main()
{
    const auto count = 8;
    const int size = count * sizeof(int);
    int h[] = {13, 65, 15, 14, 33, 2, 30, 8};

    int* d;

    cudaMalloc(&d, size);
    cudaMemcpy(d, h, size, cudaMemcpyHostToDevice);

    max1 <<<1, count / 2 >>>(d);

    int result;
    cudaMemcpy(&result, d, sizeof(int), cudaMemcpyDeviceToHost);
    // cout << "Large no is %d " << result << endl;
    printf("Large no is %d ", result);
}
```

```
getchar();

cudaFree(d);
//delete[] h;

return 0;
}
```

Large no is 4