```
#Amruta Sool
#BBCO19128
```

```
!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-ayddx_xf
  Running command git clone --filter=blob:none --quiet https://github.com/andreinechaev/nvcc4jupyter.git /tmp/pip-req-build-ayddx_xf
  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=adaf6331b3e9daf783874a365c47fbb53ca4036e30245b20da6aaa
  Stored in directory: /tmp/pip-ephem-wheel-cache-cob7hc38/wheels/a8/b9/18/23f8ef71ceb0f63297dd1903aedd067e6243a68ea756d6feea
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 min1(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, 23, 30, 8};

  int* d;

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

  min1 <<<1, count / 2 >>>(d);

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

  getchar();

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

  return 0;
 }
```

```
Small no is 8
```