DeviceQuery, bandwidthTest, and warming up the device

Exercise 1:

Please connect to the LSF front-end node, i.e. `ssh login.hpc.dtu.dk`, and from there run `hpcintrogpush` to get an interactive session on a node with a GPU, or start a ThinLinc session and open an '`xterm (hpcintrogpush)`' from the 'Applications -> DTU' menu.

In order to use the NVIDIA HPC compilers and libraries you need to load the following modules (gcc > 11.x is not supported):

```
module load nvhpc/22.11-nompi
module load cuda/11.8
module load gcc/11.3.0-binutils-2.38
```

These can be inserted into your `.bashrc` for convenience. Please note that `module load nvhpc/22.11-nompi` will also succeed on nodes without GPUs (you can work on such nodes, but running OpenMP offload code will silently default back to running on the CPU).

1. Run `nvidia-smi` to check the GPUs on your node and whether they are used.
2. Run `lscpu` and `free` to check the CPUs and size of the main memory on your node.

   Most GPU specifications are **not** available from the OpenMP API. To query more specifications we have to access them through the CUDA framework:

   Run `/appl/cuda/11.8.0/extras/demo_suite/deviceQuery`

   to see the details of the GPUs on your node. Make a note of the main differences between the CPUs and the GPUs by writing down the key specifications (core counts, clock rates, cache sizes, main memory sizes).
3. Another important specification is the bandwidth when transferring data between the CPU (host) and the GPUs (devices).

   Run `/appl/cuda/11.8.0/extras/demo_suite/bandwidthTest`

   to measure the bandwidth between host and device and from device to device. Use options `-memory=pageable` and `-memory=pinned` to see the difference between having normal vs. pinned memory (we will learn later what that means).
4. Currently it takes about 0.2 secs to warm up a GPU for first use. The file `warmUpDevice.cpp` is provided on DTU Learn. Compile this file using the `nvc++` compiler

   ```
   nvc++ -mp=gpu warmUpDevice.cpp -o warmUpDevice
   ```

   and run it. Look into the source code and familiarize yourself with it.
5. Sometimes knowing the main device specifications of your target at runtime can help tune OpenMP Offload code better. You can get the specifications in C code with the CUDA API function `cudaGetDeviceProperties()`. The file `cudaDeviceQuery.cpp` is provided on DTU Learn to show how. Compile this file using the `nvc++` compiler

   ```
   nvc++ -cuda -I /appl/nvhpc/2022_2211/Linux_x86_64/22.11/examples/OpenMP/SDK/inclu
   cudaDeviceQuery.cpp -o cudaDeviceQuery
   ```

   and run it. Look into the source code and familiarize yourself with it. More details of the `cudaDeviceProp` struct can be found here:

   `https://docs.nvidia.com/cuda/cuda-runtime-api/structcudaDeviceProp.html`