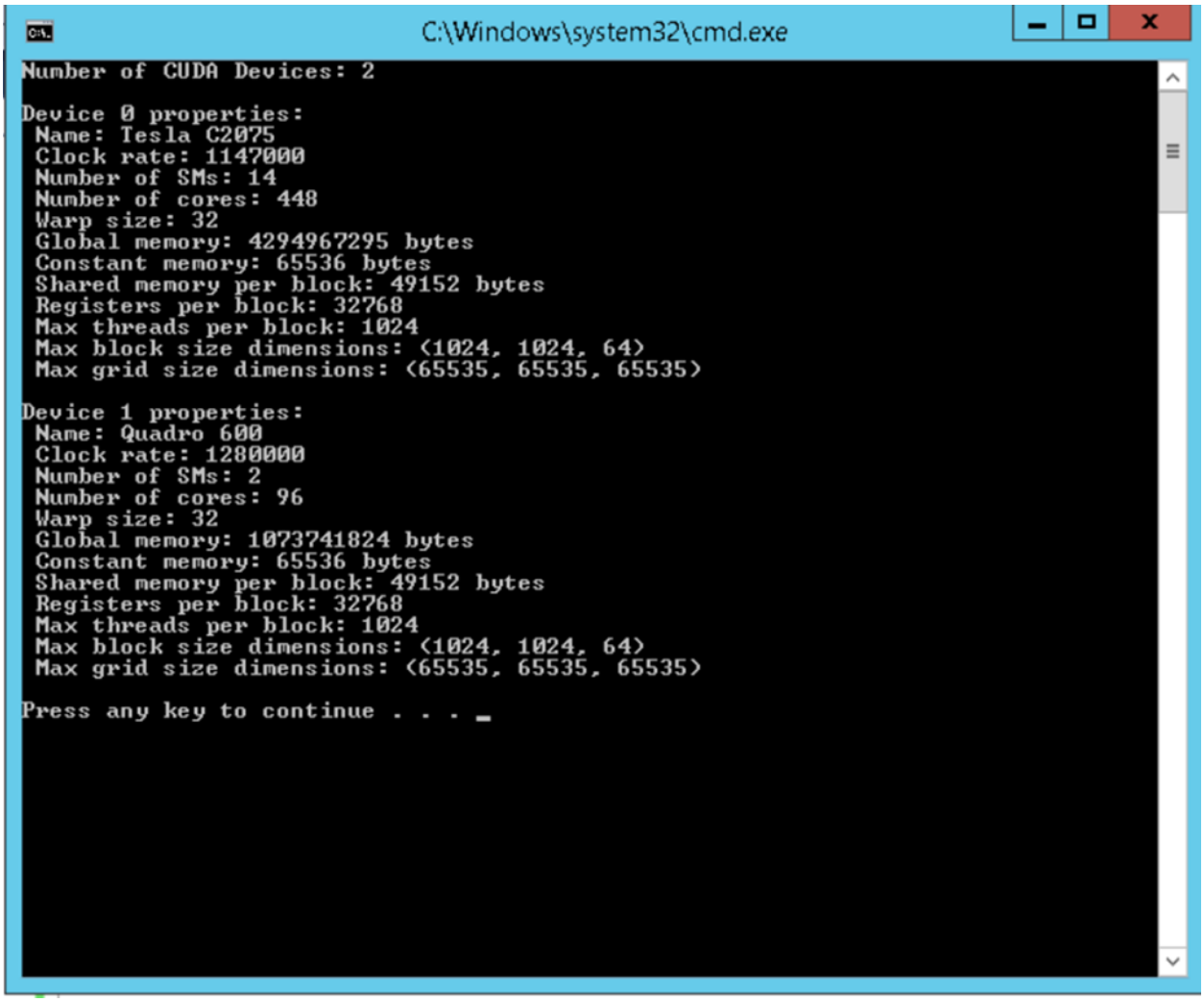
# Machine Problem 1

**Abdellah Ghassel (20230384)**

*“I do hereby verify that this machine problem submission is my own work and contains my own original ideas, concepts, and designs. No portion of this report or code has been copied in whole or in part from another source, with the possible exception of properly referenced material".*

Initialization Code and here is the terminal output:




```
/*
Name:   Abdellah Ghassel
Student #:  20230384

Note: most commands are from Professor's slides
*/

#include "cuda_runtime.h"
#include <string.h>
#include <stdio.h>
```

```
int coreCount(cudaDeviceProp dev_prop)
{
  int mpCount = dev_prop.multiProcessorCount;
  int cores = -1;

  if (dev_prop.major == 2)
  {
    cores = (dev_prop.minor == 1) ? (mpCount * 48) : (mpCount * 32);
  }
  else if (dev_prop.major == 3)
  {
    cores = mpCount * 192;
  }
  else if (dev_prop.major == 5)
  {
    cores = mpCount * 128;
  }
  else if (dev_prop.major == 6)
  {
    if (dev_prop.minor == 1 || dev_prop.minor == 2)
    {
      cores = mpCount * 128;
    }
    else if (dev_prop.minor == 0)
    {
      cores = mpCount * 64;
    }
  }
  else if (dev_prop.major == 7)
  {
    if (dev_prop.minor == 0 || dev_prop.minor == 5)
    {
      cores = mpCount * 64;
    }
  }

  return cores;
}

int main()
{
  int dev_count;
  cudaGetDeviceCount(&dev_count);
  printf("Number of CUDA Devices: %d\n\n", dev_count);
  for (int i = 0; i < dev_count; i++)
  {
    printf("Device %d properties:\n", i + 1);
    cudaDeviceProp dev_prop;
    cudaGetDeviceProperties(&dev_prop, i);

    printf(" Name: %s\n", dev_prop.name);
    printf(" Clock rate: %d\n", dev_prop.clockRate);
    printf(" Number of SMs: %d\n", dev_prop.multiProcessorCount);
    printf(" Number of cores: %d\n", coreCount(dev_prop));
    printf(" Warp size: %d\n", dev_prop.warpSize);
    printf(" Global memory: %zu bytes\n", dev_prop.totalGlobalMem);
    printf(" Constant memory: %zu bytes\n", dev_prop.totalConstMem);
    printf(" Shared memory per block: %d bytes\n", dev_prop.sharedMemPerBlock);
    printf(" Registers per block: %d\n", dev_prop.regsPerBlock);
    printf(" Max threads per block: %d\n", dev_prop.maxThreadsPerBlock);
    printf(" Max block size dimensions: (%d, %d, %d)\n", dev_prop.maxThreadsDim[0], dev_prop.maxThreadsDim[1], dev_prop.maxThreadsDim[2]);
    printf(" Max grid size dimensions: (%d, %d, %d)\n\n", dev_prop.maxGridSize[0], dev_prop.maxGridSize[1], dev_prop.maxGridSize[2]);
  }

  return 0;
}
```