A5 (10 marks)

***Focus**: CUDA (A, B) - Introduction (warming up!!)*

Q1. [+3] ***Querying your GPU***: In this question, you will run a simple query code to know the properties and limits of your NVIDIA card. Locate the "CUDA Samples" folder on your hard disk (e.g., c:\ProgramData\NVIDIA Corporation\CUDA Samples). Navigate to "1_Utilities\deviceQuery" subfolder. Then, Open the solution file (.sln) that matches your Visual Studio version. Once opened, compile and run project (Ctrl + F5). Then, capture your answers and **submit** them as an image file named A5_Q1.png.

While the above sample project provide detailed information, a simpler code (with less information) is given below. Use the below code if you cannot find or run the CUDA sample project.

Note: When creating a new project, make sure to choose CUDA template. The file extension for your CUDA program should be "cu".

Marking guide: +3 for a screenshot with the required info

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include <stdio.h>
int main(){
	cudaDeviceProp  prop;
	int count;
	cudaGetDeviceCount(&count);
	for (int i = 0; i < count; i++) {
		cudaGetDeviceProperties(&prop, i);
		printf("----- General Information for device %d ---\n", i);
		printf("Name:                          %s\n", prop.name);
		printf("Compute capability:            %d.%d\n", prop.major, prop.minor);
		printf("Clock rate:                    %d\n", prop.clockRate);
		printf("Device copy overlap:           ");
		printf(prop.deviceOverlap ? "Enabled\n" : "Disabled\n");
		printf("Kernel execution timeout: ");
		printf(prop.kernelExecTimeoutEnabled ? "Enabled\n" : "Disabled\n");
		printf("----- Memory Information for device %d ---\n", i);
		printf("Total global mem:              %lu\n", prop.totalGlobalMem);
		printf("Total constant Mem:            %ld\n", prop.totalConstMem);
		printf("Max mem pitch:                 %ld\n", prop.memPitch);
		printf("Texture Alignment:             %ld\n", prop.textureAlignment);
		printf("----- MP Information for device %d ---\n", i);
		printf("Multiprocessor count:          %d\n", prop.multiProcessorCount);
		printf("Shared mem per mp:             %ld\n", prop.sharedMemPerBlock);
		printf("Registers per mp:              %d\n", prop.regsPerBlock);
		printf("Threads in warp:               %d\n", prop.warpSize);
		printf("Max threads per block:         %d\n", prop.maxThreadsPerBlock);
		printf("Max thread dimensions:         (%d, %d, %d)\n",
			prop.maxThreadsDim[0], prop.maxThreadsDim[1],
			prop.maxThreadsDim[2]);
		printf("Max grid dimensions:           (%d, %d, %d)\n",
			prop.maxGridSize[0], prop.maxGridSize[1],
			prop.maxGridSize[2]);
		printf("\n");
	}
	return 0;
}
```

Q2. [+7] **Simple CUDA code**: consider this loop for initializing an array **a**:

```
conts int n = 10000000;              //10 millions
for (i = 0; i < n; i++)
       a[i] = (double)i / n;
```

Submit:

a) The serial implementation running on the CPU.
b) The CUDA implementation (1 thread per array element).

In both cases, add code to print the first and last 5 elements of the array to verify your code. (not that you need to use the placeholder %.7f to print 7 digits after the decimal point.

Sample output:

```
a[0]: 0.0000000
a[1]: 0.0000001
a[2]: 0.0000002
a[3]: 0.0000003
a[4]: 0.0000004
...
a[9999995]: 0.9999995
a[9999996]: 0.9999996
a[9999997]: 0.9999997
a[9999998]: 0.9999998
a[9999999]: 0.9999999
```

Marking guide:

+2 for measuring the time of the parallel and serial code
+2 for the kernel function
+3 for launch configuration and properly calling the kernel

Submission Instructions

For this assignment, you need to do the following:

1- Compress the PNG file from Q1 and the source code file (i.e. the .cu file, not the whole project) from Q2 into one zip folder and give a name to the zipped file that matches your ID (e.g., 1234567.zip).
2- Submit the zipped file to Canvas.

Note that you can resubmit an assignment, but the new submission overwrites the old submission and receives a new timestamp.