EN605.617 Module 5: Stretch Problem

Opinions given on page 3.

```
//=============================================================================
// copyToShared: Copys our pined data to shared memory on the device
//=============================================================================
__device__ void copyToShared(const int * const global_a, const int * const
global_b, int * shared_a,int * shared_b) {
      int tid = threadIdx.x + blockIdx.x*blockDim.x;
      for (int i = 0; i < N; i++) {
            //Not giving anything away here but presume this was written well
      }
      __syncthreads();
}

//=============================================================================
// add: Just adds the arrays that are passed to it.
//=============================================================================
__device__ void add(int *a, int *b, int *c) {
      //Not giving anything away here but presume this was written well
}

//=============================================================================
// run_gpu: Uses a switch statement for our modules. Runs the same exact
// add program on the device but with the data in different
// types of memory.
//=============================================================================
__global__ void run_gpu(int *a, int *b, int *c, int Module) {
      switch (Module)
      {
            case 5:
                  //presume that shared memory is allocated correctly
                  copyToShared(a,b,shared_a,shared_b);
                  add(shared_a, shared_b, shared_c);
                  break;
            default:
                  break;
      }
}

int main(int argc,char* argv[]) {
      outputCardInfo();
      const unsigned int bytes = N * sizeof(int);
      //Host located pageable
      int a[N], b[N], c[N];
      //To be pinned memory
      int *h_pa, *h_pb, *h_pc;
      //Device global mem holders
```

```
	int *dev_a, *dev_b, *dev_c;

	//Allocate devices - Presumed that below is done correctly
	...

	//Allocate pinned - Presumed that below is done correctly
	...

	//Populate our arrays with numbers.  - Presumed that below is done correctly
	for (int i = 0; i < N; i++) {
		...
	}


	//============================================================================
	// Module 5: Show difference between Shared and Constant
	//============================================================================
	printf("\n\nModule 5:");
	printf("\nKernel using Pinned Memory copied to Shared:\n");
	speedTest(h_pa, h_pb, h_pc, dev_a, dev_b, dev_c, bytes, 5);
	printf("\nKernel using Pinned Memory copied to Constant:\n");
	speedTest(h_pa, h_pb, h_pc, dev_a, dev_b, dev_c, bytes, 6);

	//What is missing from above?

	//Free host and device memory - Presumed that below is done correctly
	...

	return 0;
}

//============================================================================
// SpeedTest: Returns speed of executing the kernel
//============================================================================
void speedTest(int *a, int *b, int *c, int *deva, int *devb, int *devc,int bytes,
int Module) {
	// copy from host memory to device - Presumed that below is done correctly
	auto start = std::chrono::high_resolution_clock::now();
	run_gpu <<<16, 256 >>> (deva, devb, devc, Module);
	auto stop = std::chrono::high_resolution_clock::now();

	// copy results from device memory to host - Presumed that below is done
correctly
	printf("Kernel finished in:%d\n", stop - start);
}
```

The previous code has logic errors included in it’s design.

For example, the speedTest() method doesn’t have a function prototype and it’s used before it’s defined.

The run_gpu<<< >>>() kernel will never run the contant memory addition kernel add<<< >>>() since the switch statement in run_gpu<<< >>>() will break if anything other that 5 is passed. This means that:

```
speedTest(h_pa, h_pb, h_pc, dev_a, dev_b, dev_c, bytes, 6);
```

will never produce anything.

The switch statement to run the different types of tests shouldn’t be in a gpu kernel, the kernel run_gpu<<< >>>() should be a function on the host machine.

The copyToShared<<< >>>() kernel doesn’t declare any shared memory, whether it is dynamic or static shared memory. If it’s intended to be dynamic then the run_gpu<<< >>>() kernel doesn’t pass any indication of bytes in shared memory.

In the speedTest() method there is no code copying over samples from host memory into the global memory in the GPU. In essence no samples are being processed.

There is not implementation of the outputCardInfo() method.