PartA

Problem1:

Q1:

```
(habitat) Singularity> ./vecadd00 500
Total vector size: 3840000
Time: 0.000331 (sec), GFlopsS: 11.603838, GBytesS: 139.246058
Test PASSED
(habitat) Singularity> ./vecadd00 1000
Total vector size: 7680000
Time: 0.000589 (sec), GFlopsS: 13.041399, GBytesS: 156.496784
Test PASSED
(habitat) Singularity> ./vecadd00 2000
Total vector size: 15360000
Time: 0.001138 (sec), GFlopsS: 13.497697, GBytesS: 161.972368
Test PASSED
```

Q2:

```
(habitat) Singularity> ./vecadd01 500
Total vector size: 3840000
Time: 0.000008 (sec), GFlopsS: 488.064465, GBytesS: 5856.773585
Test FAILED
(habitat) Singularity> ./vecadd01 1000
Total vector size: 7680000
Time: 0.000008 (sec), GFlopsS: 947.419256, GBytesS: 11369.031078
Test FAILED
(habitat) Singularity> ./vecadd01 2000
Total vector size: 15360000
Time: 0.000008 (sec), GFlopsS: 1894.838513, GBytesS: 22738.062155
Test FAILED
```

It appears that `vecadd01` consistently fails with shorter execution times but significantly larger values for GFlopsS and GBytesS.

Problem2:

Q3:

```
(habitat) Singularity> ./matmult00 256
Data dimensions: 4096x4096
Grid Dimensions: 256x256
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 0.089620 (sec), nFlops: 137438953472, GFlopsS: 1533.572636
(habitat) Singularity> ./matmult00 512
Data dimensions: 8192x8192
Grid Dimensions: 512x512
```

```
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 0.690086 (sec), nFlops: 1099511627776, GFlopsS: 1593.296787


TEST FAILED: number of errors:  4321886, max rel error: 0.000122
(habitat) Singularity> ./matmult00 1024
Data dimensions: 16384x16384
Grid Dimensions: 1024x1024
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 5.569285 (sec), nFlops: 8796093022208, GFlopsS: 1579.393612


TEST FAILED: number of errors:  138340737, max rel error: 0.000244
(habitat) Singularity>
```

Q4：

```
(habitat) Singularity> ./matmult00 256
Data dimensions: 8192x8192
Grid Dimensions: 256x256
Block Dimensions: 32x32
Footprint Dimensions: 32x32
Time: 0.554000 (sec), nFlops: 1099511627776, GFlopsS: 1984.678384


TEST FAILED: number of errors:  4321886, max rel error: 0.000122
(habitat) Singularity> ./matmult00 512
Data dimensions: 16384x16384
Grid Dimensions: 512x512
Block Dimensions: 32x32
Footprint Dimensions: 32x32
Time: 4.439757 (sec), nFlops: 8796093022208, GFlopsS: 1981.210521


TEST FAILED: number of errors:  138340737, max rel error: 0.000244
(habitat) Singularity> ./matmult00 1024
Data dimensions: 32768x32768
Grid Dimensions: 1024x1024
Block Dimensions: 32x32
```

```
Footprint Dimensions: 32x32
Time: 35.582783 (sec), nFlops: 70368744177664, GFlopsS: 1977.606522


TEST FAILED: number of errors:  849488718, max rel error: 0.000488
(habitat) Singularity>
```

Q5：
The experiments show that optimizing memory access patterns and choosing appropriate thread block sizes significantly improve CUDA program performance, while highlighting the importance of numerical stability and correctness considerations.

Part-B: CUDA Unified Memory (20 points)
In this problem we will compare vector operations executed on host vs on GPU to quantify the speed-up.
Q1 4 points Write a C++ program that adds the elements of two arrays with a K million elements each. Here K is a parameter. Profile and get the time to execute this program for K=1,5,10,50,100. Use free() to free the memory at the end of the program.

```
(habitat) Singularity> g++ -o Q1 Q1.cpp
(habitat) Singularity> ./Q1
Time to execute for K = 1 million: 4 ms
Time to execute for K = 5 million: 20 ms
Time to execute for K = 10 million: 41 ms
Time to execute for K = 50 million: 205 ms
Time to execute for K = 100 million: 409 ms
```

Q2：

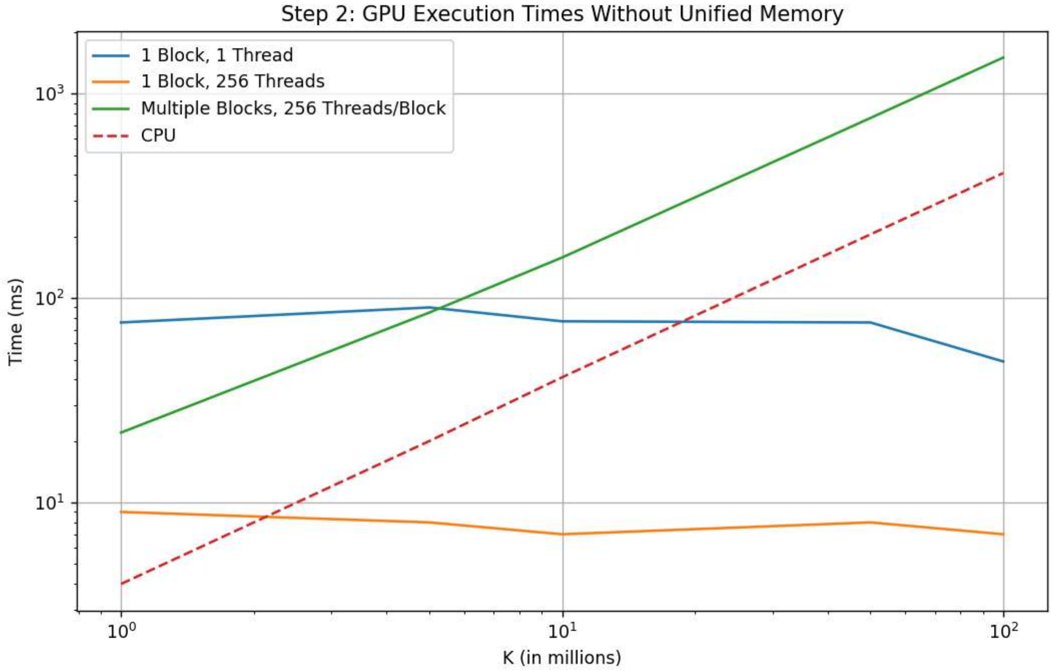
```
(habitat) Singularity> ./Q2
K = 1, 1 Block, 1 Thread, Time: 76 ms
K = 1, 1 Block, 256 Threads, Time: 9 ms
K = 1, 3907 Blocks, 256 Threads/Block, Time: 22 ms
K = 5, 1 Block, 1 Thread, Time: 90 ms
K = 5, 1 Block, 256 Threads, Time: 8 ms
K = 5, 19532 Blocks, 256 Threads/Block, Time: 85 ms
K = 10, 1 Block, 1 Thread, Time: 77 ms
K = 10, 1 Block, 256 Threads, Time: 7 ms
K = 10, 39063 Blocks, 256 Threads/Block, Time: 158 ms
K = 50, 1 Block, 1 Thread, Time: 76 ms
K = 50, 1 Block, 256 Threads, Time: 8 ms
K = 50, 195313 Blocks, 256 Threads/Block, Time: 760 ms
```

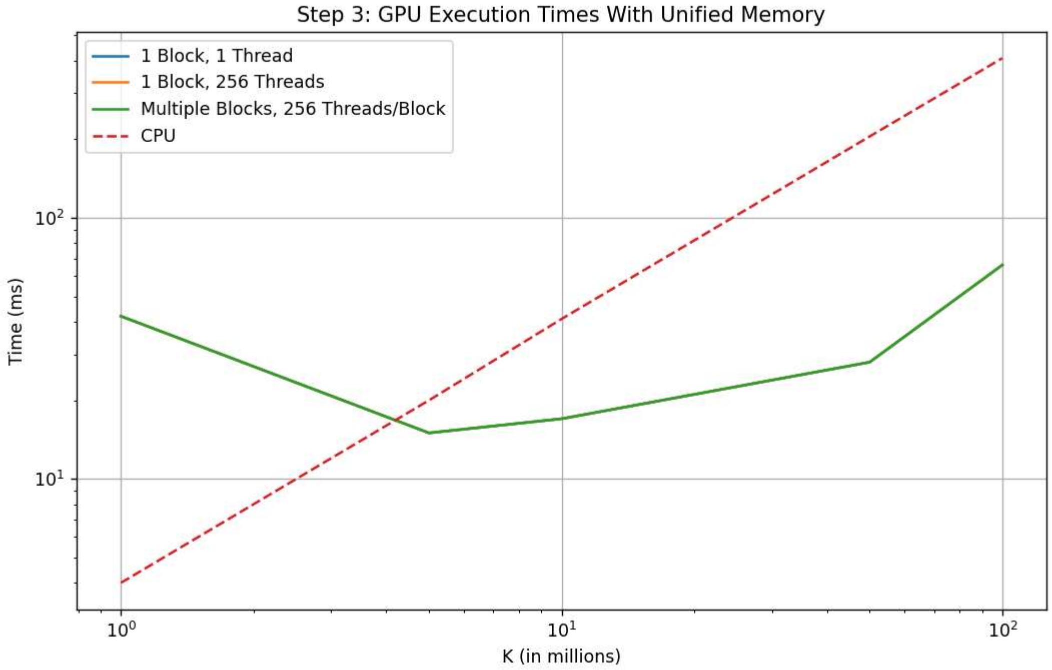
```
K = 100, 1 Block, 1 Thread, Time: 49 ms
K = 100, 1 Block, 256 Threads, Time: 7 ms
K = 100, 390625 Blocks, 256 Threads/Block, Time: 1501 ms

(habitat) Singularity>
```

Q3:

```
(habitat) Singularity> ./Q3
Time to execute for K = 1 million: 42 ms
Time to execute for K = 5 million: 15 ms
Time to execute for K = 10 million: 17 ms
Time to execute for K = 50 million: 28 ms
Time to execute for K = 100 million: 66 ms
```

Q4：





Part-C: Convolution in CUDA (40 points)

C1:

```
(habitat) Singularity> ./c1
```

```
Kernel execution time: 11.012 milliseconds
Checksum: 122756344698240.000
```

C2：

```
(habitat) Singularity> ./c2
Tiled convolution kernel execution time: 1.925 milliseconds
Checksum of the tiled output tensor O: 122756344698240.000
```

C3：

```
(habitat) Singularity> ./c3
Iteration 0: Convolution execution time: 8015.012 milliseconds
Iteration 1: Convolution execution time: 13.058 milliseconds
Iteration 2: Convolution execution time: 13.016 milliseconds
Iteration 3: Convolution execution time: 12.997 milliseconds
Iteration 4: Convolution execution time: 12.993 milliseconds
Checksum: 122756344698240.000
```