Vector Add

```
[junseo@r40a-09.sif bin]$ ./VectorAdd_Solution -e ../VectorAdd/Dataset/0/output.raw -i ../VectorAdd/Dataset/0/input0.raw,../VectorAdd/Dataset/0/in
put1.raw \ -o result0.raw -t vector
[TIME][Generic][Importing data and creating memory on host][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module3/VectorAdd/solution.cu: 24-30] E
lapsed time: 11.7298 ms
The input length is 16
[TIME][GPU][Allocating GPU memory.][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module3/VectorAdd/solution.cu: 34-41] Elapsed time: 0.230177 ms
[TIME][GPU][Copying input memory to the GPU.][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module3/VectorAdd/solution.cu: 43-49] Elapsed time: 0
.04006 ms
[TIME][Compute][Performing CUDA computation][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module3/VectorAdd/solution.cu: 56-66] Elapsed time: 0.
544428 ms
[TIME][Copy][Copying output memory to the CPU][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module3/VectorAdd/solution.cu: 68-72] Elapsed time:
0.029234 ms
[TIME][GPU][Freeing GPU Memory][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module3/VectorAdd/solution.cu: 74-80] Elapsed time: 0.145005 ms
The solution is correct
[junseo@r40a-09.sif bin]$ ./VectorAdd_Solution -e ../VectorAdd/Dataset/0/output.raw -i ../VectorAdd/Dataset/0/input0.raw,../VectorAdd/Dataset/0/in
put1.raw \ -o result0.raw -t vector
[TIME][Generic][Importing data and creating memory on host][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module3/VectorAdd/solution.cu: 24-30] E
lapsed time: 11.441 ms
The input length is 16
[TIME][GPU][Allocating GPU memory.][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module3/VectorAdd/solution.cu: 34-41] Elapsed time: 0.376009 ms
[TIME][GPU][Copying input memory to the GPU.][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module3/VectorAdd/solution.cu: 43-48] Elapsed time: 0
.03449 ms
[TIME][Compute][Performing CUDA computation][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module3/VectorAdd/solution.cu: 54-64] Elapsed time: 0.
453247 ms
[TIME][Copy][Copying output memory to the CPU][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module3/VectorAdd/solution.cu: 66-70] Elapsed time:
0.072061 ms
[TIME][GPU][Freeing GPU Memory][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module3/VectorAdd/solution.cu: 72-78] Elapsed time: 0.148419 ms
The solution is correct
```

Figure 1: `VectorAdd_Solution` output

Questions

1. How many floating operations are being performed in your vector add kernel? EXPLAIN.

   The kernel perfroms `inputLength` floating point operations. This is because the if statement in the kernel function `index < len` limits the number of threads that are actually performing the floating point operations.

2. How many global memory reads are being performed by your kernel? EXPLAIN.

   The kernel performs `inputLength * 2` global memory reads because each thread reads an element each from the two input arrays.

3. How many global memory writes are being performed by your kernel? EXPLAIN.

   The kernel performs `inputLength` global memory writes because each thread writes to the output arrary in the if statement.

4. Describe what possible optimizations can be implemented to your kernel to achieve a performance speedup.

   We can achieve a performance speedup by using block sizes that are a multiple of the warp size (32) and using shared memory to reduce the number of global memory reads and writes.

5. Name three applications of vector addition.

   i Image processing (e.g. blur, translating an image on a screen)

   ii Phyics simulations (e.g. N-body simulations require vector addition to find the net force and momenta of N-particles)

   iii Single value decomposition (SVD) (e.g. crude image compression [web demo] uses SVD which requires matrix multiplication and vector addition)