Roll Numbers:

Asjad Siddiqui - 22L-6602
Qainat Saeed - 22L-6569

Part 1:

Output:

Found 1 CUDA devices Device 0: NVIDIA GeForce RTX 2070 Super SMs: 40 Global mem: 8192 MB CUDA Cap: 7.5
---------- Running 3 timing tests: CUDA saxpy kernel time: 7.977 ms [140.095 GB/s] Effective BW by CUDA saxpy: 390.048 ms [2.865 GB/s] CUDA saxpy kernel time: 4.571 ms [244.473 GB/s] Effective BW by CUDA saxpy: 253.462 ms [4.409 GB/s] CUDA saxpy kernel time: 4.311 ms [259.255 GB/s] Effective BW by CUDA saxpy: 249.102 ms [4.486 GB/s]

Question 1:

the GPU implementation is much faster than a CPU version would be. CUDA kernel runs in just 4-8 milliseconds, which is very quick compared to what a CPU could do. This is because GPUs have thousands of cores that can work on the simple SAXPY calculation all at once, while CPUs have far fewer cores.

Question 2:

big difference between:

- Just the kernel time (4-8ms)
- The total time including transfers (249-390ms)

The difference shows that moving data between CPU and GPU is the real bottleneck. Kernel reaches impressive speeds (140-259 GB/s) when just looking at computation, but the overall process is much slower (only 2.9-4.5 GB/s) because of the time it takes to copy data back and forth.

Part 2:

Found 1 CUDA devices
Device 0: NVIDIA GeForce RTX 2070 Super
SMs: 40

```
   Global mem: 8192 MB
   CUDA Cap:   7.5
----------------------------------------------------------
Array size: 64
Student GPU time: 1.769 ms
Scan outputs are correct!
```

Question scan

iterative upsweep and downsweep algorithm as shown in the pseudo-code was followed

Part 3:

```
Found 1 CUDA devices
Device 0: NVIDIA GeForce RTX 2070 Super
   SMs:        40
   Global mem: 8192 MB
   CUDA Cap:   7.5
----------------------------------------------------------

Running benchmark, 1 frames, beginning at frame 0 ...
Dumping frames to output_xxx.ppm
Copying image data from device
Wrote image file output_0000.ppm
Clear:     0.1602 ms
Advance:   0.0007 ms
Render:    0.1360 ms
Total:     0.2969 ms
File IO:   1027.6496 ms

Overall:  1.0280 sec (note units are seconds)
```

Question Render

In our pixel-based CUDA implementation, we assigned each thread to process one pixel in the final image. This approach solved the race condition problem

from the original implementation.

For each pixel, we process all circles in their original input order which is essential for correct blending. We calculate the distance between each pixel and circle center to determine if the pixel falls within the circless boundary.

We handle snowflake scenes differently from other scenes applying appropriate color lookup and alpha calculations based on the normalized distance from the circle center.

our approach is that it eliminates race conditions since each pixel is processed by exactly one thread.
our implementation maintains proper rendering order and transparency blending having both requirements while keeping the CUDA code relatively simple