List Scan

Questions

1. 3 Applications of parallel scan:
   (a) Radix sort or counting sorting
   (b) Summed area table (SAT) for image processing (e.g. depth of field blurring)
   (c) numerical integration (e.g. trapezoidal rule or Simpson's rule)

2. There are
$$N/2 + N/4 + \ldots + 1 = N - 1$$
$n - 1$ additions in the up-sweep phase and
$$\sum_{i=1}^{\log_2 n} \frac{n}{2^i} - 1 = n - 1 - \log_2 n$$
$n - 1 - \log_2 n$ additions in the down-sweep phase for a total of $2n - 2 - \log_2 n$ additions.

3. The kernel has $2 \times$ `blockDim.x` global reads for the shared memory step since each thread reads 2 elements. So we have $2 \times$ `blockDim.x` $\times$ `numBlocks` global reads.

4. Just like the global reads, the writes are done once per block so we have $2 \times$ `blockDim.x` $\times$ `numBlocks` global writes.

5. MAX: A thread (the first thread) would have at max $2 \times \log_2 N - 1$ additions in total ($\log_2 N - 1$ per phase).
   MIN: A thread (the last) would have at min 1 addition in the up-sweep phase and 0 additions in the down-sweep phase or 1 addition in total.
   AVERAGE: The average number of operations is then
$$\frac{2 \times \log_2 N - 2 + 1}{N}$$

6. In the up-sweep the stride doubles from 1 to 512 or $\log_2 512 + 1 = 10$ times, and in the down-sweep the stride halves from $512/2 = 256$ to 1 or $\log_2 256 + 1 = 9$ times and once again after the last iterations so we have $10 + 9 + 1 = 20$ synchronizations in total for a block size of 512.

7. We used shared memory to reduce the number of global memory accesses and also stored the sum of the block in the same kernel to reduce the number of kernel launches. Also

8. We can implement thread coarsening and a single pass-scan (doing the segemented scan all in one kernel) to reduce the memory accesses.

9. We have implemented a recursive segemented scan algorithm to handle an arbitrary number of elements! (it works with smaller block sizes up to dataset 12 but 13 has some floating point errors propogating)

10. if the order matters, we can use the Kogge-Stone scan algorithm for non-commutative operations.

11. We can get different results because of the floating point errors that are non-associative.

OUPUT

```
[junseo@r40a-09.sif ListScan]$ ./listscan.sh
     0  The solution is correct
     1  The solution is correct
     2  The solution is correct
     3  The solution is correct
     4  The solution is correct
     5  The solution is correct
     6  The solution is correct
     7  The solution is correct
     8  The solution is correct
     9  The solution is correct
    10  The solution is correct
    11  The solution is correct
    12  The solution is correct
```

Figure 1: Successes