```
//Implement Min, Max, Sum and Average oprations using Parallel Reduction.


#include <iostream>
#include <cuda.h>

__global__ void reduceMinMaxSum(int *a, int *mn, int *mx, int *sm) {
    __shared__ int smin[256], smax[256], ssum[256];
    int t = threadIdx.x;
    smin[t] = smax[t] = ssum[t] = a[t];
    __syncthreads();
    for (int s = 128; s > 0; s >>= 1) {
        if (t < s) {
            smin[t] = min(smin[t], smin[t + s]);
            smax[t] = max(smax[t], smax[t + s]);
            ssum[t] += ssum[t + s];
        }
        __syncthreads();
    }
    if (t == 0) *mn = smin[0], *mx = smax[0], *sm = ssum[0];
}

int main() {
    int h[256], *d, *mn, *mx, *sm, x, y, z;
    for (int i = 0; i < 256; i++) h[i] = rand() % 100;

    cudaMalloc(&d, 256 * 4); cudaMalloc(&mn, 4); cudaMalloc(&mx, 4);
cudaMalloc(&sm, 4);
    cudaMemcpy(d, h, 256 * 4, cudaMemcpyHostToDevice);
    reduceMinMaxSum<<<1, 256>>>(d, mn, mx, sm);
    cudaMemcpy(&x, mn, 4, cudaMemcpyDeviceToHost);
    cudaMemcpy(&y, mx, 4, cudaMemcpyDeviceToHost);
    cudaMemcpy(&z, sm, 4, cudaMemcpyDeviceToHost);

    std::cout << "Min: " << x << "\nMax: " << y << "\nSum: " << z << "\nAvg: "
<< z / 256.0f << "\n";
    cudaFree(d); cudaFree(mn); cudaFree(mx); cudaFree(sm);
    return 0;
}

//nvcc -o reduction reduction.cu
//./reduction
```