Problem 1. Show that if a sequence $X = x_0, \ldots, x_{w-1}$ is $k$-smooth, then the result of passing $X$ through a balancing network is $k$-smooth.
Problem 2. Let $\mathcal{B}$ be a width-$w$ balancing network of depth $d$ in a quiescent state $s$. Let $n = 2^d$. Prove that if $n$ tokens enter the network on the same wire, pass through the network, and exit, then $\mathcal{B}$ will have the same state after the tokens exit as it did before they entered.
Problem 3. Consider the following code for an in-place merge-sort:

```java
void mergeSort(int[] A, int lo, int hi) {
    if (hi > lo) {
        int mid = (hi - lo)/2;
        executor.submit(new mergeSort(A, lo, mid));
        executor.submit(new mergeSort(A, mid+1, hi));
        awaitTermination();
        merge(A, lo, mid, hi);
    }
}
```

Assuming that the merge method has no internal parallelism, give the work, span, and parallelism of this algorithm. Give your answers both as recurrences and as $\Theta(f(n))$, for some function $f$. 
Problem 4. Professor Jones takes some measurements of his (deterministic) multithreaded program, which is scheduled using a greedy scheduler, and finds that $T_4 = 80$ seconds and $T_{64} = 10$ seconds. What is the fastest that the professor’s computation could possibly run on 10 processors? Use the following inequalities and the bounds implied by them to derive your answer. Note that $P$ is the number of processors and $T_P$ is the time taken with $P$ processors.

\begin{align*}
T_P & \geq \frac{T_1}{P} \quad (0.1) \\
T_P & \geq T_\infty \quad (0.2) \\
T_P & \leq \frac{(T_1 - T_\infty)}{P} + T_\infty \quad (0.3)
\end{align*}

(The last inequality holds on a greedy scheduler.)
Problem 5. Programming Assignment:
For this programming assignment, you will need to implement the Bitonic Counting network described in Chapter 12 of the book so that it supports both traversals (used by tokens) and anti-traversals (used by anti-tokens). Recall that given the same Balancer state an anti-traversal will send its anti-token on the wire opposite to that a normal traversal would send its own token.

The stencil code for this assignment can be found in /course/cs1760/pub/bitonic. The files you will need to modify are Bitonic.java, Merger.java, and Balancer.java. A stencil of a tester class as well as a class that houses the main() function is also provided in the form of TestBitonic.java and BitonicTester.java, the both of which you can edit to test your implementations. An optional constants class in Consts.java is also provided for you to use at your own discretion.