For each algorithm that you are asked to write/develop, you should provide arguments similar to paragraphs titled, **Correctness** and **Analysis** in Section 30.1.1. Specifically you must argue why the program that you wrote computes the correct answer, what its time complexity is and what model of PRAM it operates on.

**Problem 1, List Reversal:**

An $n$-node list is stored in the memory of a PRAM, and there are two additional variables, `head` (initialized to the index of the head of the list) and `tail` which is uninitialized.

**Q1:** Write the fastest algorithm (you will lose points if the algorithm is not fast enough) to reverse the list, and also reset the variables `head` and `tail` appropriately, relative to the new list. Provide an argument whether or not it is EREW. Analyze its execution time and work. State, with a justification, whether or not it is work efficient.

**Problem 2, Concurrent Writes:**

**Q2.a:** Write an CRCW program (using the common model to resolve conflicts during concurrent writes) that computes the smallest of $n$ numbers in constant time.

**Q2.b:** Is this algorithm work efficient? Justify: if so explain why, and if not by what factor should its work be reduced in order to make it work efficient?

**Q2.c:** Write the most efficient work-efficient EREW algorithm to solve this problem (you may have to use Brent’s theorem).

**Q2.d:** Write the fastest work-efficient ERCW algorithm (pick the most favorable choice of conflict resolution) to solve this problem.

**Problem 3, Array prefix:**

**Q3.a:** Write the pseudo-code for the divide-and-conquer program to compute the prefix min of an array of $N$ numbers (assume that $N$ is a power of 2), where each step of the recursion calls two subproblems of size $N/2$ (the first array-based algorithm developed in class). You may write this code with a recursion or you may “unroll” the recursion and write a loop program.

**Q3.b:** Analyze the execution time and work of your algorithm.

**Q3.c:** Modify it so that it can execute on a $p$-processor CREW PRAM, for any value of $p$ less than $N$ (assume that $p$ is also a power of 2). For what value of $p$ is this algorithm work-efficient?