Homework Assignment #1
Due: September 12, 2025 at 5:00 p.m.

Before working on this assignment, you should read the course policy on academic honesty on the course web page at www.eecs.yorku.ca/course/4171.

1. In class, we saw a parallel algorithm for computing prefix sums. Given an array $A[1..n]$ it computes an array $B[1..n]$ such that $B[i] = A[1] + A[2] + \cdots + A[i]$. It works on an EREW PRAM in $O(\log n)$ time using $O(n)$ work.

Exactly the same algorithm would work for some other binary operators. Let $\odot$ be some binary operator on elements of array A and suppose we want to compute the array B so that $B[i] = A[1] \odot A[2] \odot \cdots \odot A[i]$.

Formally, if the elements of A are drawn from the set S, then a binary operator $\odot$ is a function that maps pairs $(x, y) \in S \times S$ to an element of S, denoted $x \odot y$. Assume expressions with several copies of the operator are evaluated left to right. For example, $x \odot y \odot z$ denotes $(x \odot y) \odot z$.

(a) Give an example of a binary operator $\odot$ (other than addition) for which the prefix sum algorithm works.

(b) Give an example of a binary operator $\odot$ for which the prefix sum algorithm does *not* work.

(c) What property or properties must the operator $\odot$ have to ensure that the prefix sum algorithm works?

(d) Suppose you have an array $A[1..n]$ and you want to copy the value in $A[1]$ into all other entries of the array. (Notice that this problem is not trivial in the EREW PRAM model.) Define a simple binary operator $\odot$ such that prefix sum achieves this copying. Explain why your operator satisfies part (c).

(e) Let p be a polynomial of degree d on the integers modulo N:

$$p(x) = a_d x^d + a_{d-1} x^{d-2} + \cdots a_1 x + a_0,$$

where x and all coefficients a_i are integers between 0 and $N - 1$ and all arithmetic is done modulo N. (Here, N is small enough to fit into one word of memory so using modular arithmetic means you never have to worry about overflow errors.) Describe an EREW PRAM algorithm that takes x as input and computes $p(x)$, and explain why your algorithm works. Assume a processor can add or multiply two numbers modulo N in one step. Give good upper bounds on the time and work used by your algorithm.