CX 4220/CSE 6220 High Performance Computing
Homework 2 Solutions

1. (a) $(a \oplus b) \oplus c = (2a + b) \oplus c = 2(2a + b) + c = 4a + 2b + c$.
   $a \oplus (b \oplus c) = a \oplus (2b + c) = 2a + 2b + c$.
   The operation $\oplus$ is not associative. It cannot be used with parallel prefix.

   (b) $(a \oplus b) \oplus c = \sqrt{a^2 + b^2} \quad \oplus \quad c = \sqrt{a^2 + b^2 + c^2}$.
   $a \oplus (b \oplus c) = a \quad \oplus \quad \sqrt{b^2 + c^2} = \sqrt{a^2 + b^2 + c^2}$.
   The operation $\oplus$ is associative. It can be used with parallel prefix.

2. (a) Compute parallel prefix on A using max as the operator and store the results in S. Modify the parallel prefix algorithm so that current number is not included in prefix sum (since we want to know the max. height of trees that occur before, not including the current one).

   (b) for each i: if $S[i] \geq A[i]$, set $A[i]$ to zero.

   (c) Sum the number in Array A (using "finding the sum" algorithm described in class). P_0 has the total score.

   Adding the parallel run-times of the three individual steps, total
   Computation time: $O\left(\frac{n}{p} + \log p\right)$
   Communication time: $O\left((\tau + \mu) \log p\right)$

3. (a) Create array A of length n such that +1 is used for '(' and -1 is used for ')'.

   (b) Compute prefix sums on A and store the result in S. Note that the sequence is well-formed if the last element of S is 0 and none of the elements in S is negative.

   (c) Create array B of length n such that $\forall 0 \leq i < n - 1$, $B[i] = 1$ if $S[i] < 0$ and $B[i] = 0$ otherwise. Also, set $B[n-1] = 1$ if $S[n-1] \neq 0$, and $B[n-1] = 0$ otherwise. Essentially, a '1' in any position in B indicates the sequence is not well-formed.

   (d) Use "finding the sum" algorithm on B with logical OR operation. The result is on P_0.

   (e) If the result is 0, the sequence is well-formed. If the result is 1, it is not well-formed.

   Adding the parallel run-times of the individual steps, total
   Computation time: $O\left(\frac{n}{p} + \log p\right)$
   Communication time: $O\left((\tau + \mu) \log p\right)$

4. (a) Create arrays $R_0 = \bar{L}$ and $R_1 = L$, where $\bar{L}$ denotes inverting each boolean value in L. Essentially, R_0 contains 1 for each item with label 0, and R_1 contains 1 for each item with label 1.

   (b) Run parallel prefix and total sum on both R_0 and R_1.

(c) If $L[i] = 0$, rank of $A[i]$ is the prefix sum result at position i of the prefix sums computed on R_0.
If $L[i] = 1$, rank of $A[i]$ is the total sum of R_0 plus the prefix sum result at position i of the prefix sums computed on R_1.

Adding the parallel run-times of the individual steps, total
Computation time: $O\left(\frac{n}{p} + \log p\right)$
Communication time: $O\left((\tau + \mu) \log p\right)$

5. Consider the following identity:

$$\begin{bmatrix} S_i & 1 \end{bmatrix} = \begin{bmatrix} S_{i-1} & 1 \end{bmatrix} \begin{bmatrix} 1 - B[i] & 0 \\ X[i] & 1 \end{bmatrix}$$

Therefore, compute parallel prefix on matrices

$$\begin{bmatrix} 1 - B[1] & 0 \\ X[1] & 1 \end{bmatrix}, \begin{bmatrix} 1 - B[2] & 0 \\ X[2] & 1 \end{bmatrix}, \begin{bmatrix} 1 - B[3] & 0 \\ X[3] & 1 \end{bmatrix}, \ldots, \begin{bmatrix} 1 - B[n-1] & 0 \\ X[n-1] & 1 \end{bmatrix}$$

using matrix multiplication as the operator, and use the prefix matrix products to compute $S[1], \ldots, S[n-1]$. The run-time is the same as parallel prefix.
Computation time: $O\left(\frac{n}{p} + \log p\right)$
Communication time: $O\left((\tau + \mu) \log p\right)$