Homework Assignment #4
Due: October 10, 2025 at 11:59 p.m.

1. We use the following approach for adding two n-bit numbers x and y suggested by a student in class. (For simplicity, assume n is a power of 2.) Divide x and y into two halves of $n/2$ bits each, add the high-order halves recursively, add the low-order halves recursively, and combine the results.

One difficulty in combining the results is that the high-order bits of the sum depend upon whether there is a carry from the low-order half. So, we generalize the problem a little: instead of returning one result, the addition routine will output *two* results. The first will be the sum of the two numbers, and the second will be the sum of the two numbers plus 1.

[7] **(a)** Assume the numbers are given as arrays of bits $X[0..n-1]$ and $Y[0..n-1]$ so that $x = \sum_{i=0}^{n-1} X[i]2^i$ and $y = \sum_{i=0}^{n-1} Y[i]2^i$. Fill in the missing parts of the pseudocode below so that the returned arrays $Z0$ and $Z1$ satisfy the postconditions $x+y = \sum_{i=0}^{n} Z0[i]\cdot 2^i$ and $x+y+1 = \sum_{i=0}^{n} Z1[i]\cdot 2^i$.

```
1:  ADD(array[0..n−1] of bit X, array[0..n−1] of bit Y) : ⟨array[0..n] of bit, array[0..n] of bit⟩
2:      if n = 1 then
3:          return _____
4:      else
5:          copy X[0..n/2 − 1] into X_0[0..n/2 − 1]          ▷ low-order bits of X
6:          copy X[n/2..n − 1] into X_1[0..n/2 − 1]          ▷ high-order bits of X
7:          copy Y[0..n/2 − 1] into Y_0[0..n/2 − 1]          ▷ low-order bits of Y
8:          copy Y[n/2..n − 1] into Y_1[0..n/2 − 1]          ▷ high-order bits of Y
9:          for j = 0, 1 in parallel do
10:             ⟨Z0_j, Z1_j⟩ ← ADD(X_j, Y_j)
11:         ▷ Describe below how to compute Z0[0..n] and Z1[0..n] from Z0_0, Z1_0, Z0_1, Z1_1
12:         ▷ n EREW processors should be able to compute Z0 and Z1 in O(1) time.
13:
14:
15:
16:
17:
18:
19:         return ⟨Z0, Z1⟩
```

[3] **(b)** Analyze the time and work needed by the algorithm. How does it compare to the addition algorithm in class?

[3] **(c)** Instead of returning two results, we could instead just return $Z0$. Then, if $Z1$ is needed (for example on lines 13–18), we could compute it from $Z0$. Describe how to compute $Z1[0..n]$ from $Z0[0..n]$ in $O(1)$ time using n processors on a PRIORITY CRCW PRAM.
Hint: think about the algorithm for adding 1 to a binary number.

In practice, we would not do so much copying of arrays; for example on lines 5–8 we would just pass indices that tell us which part of the arrays X and Y are being passed to the recursive calls. The presentation above is just for conceptual simplicity.