11. (extra credit) 1.37.

12. (extra credit) This problem is concerned with the efficient parallel computation to do binary addition of two \( n \)-bit integers \( x \) and \( y \).

Let \( x = a_{n-1}a_{n-2} \ldots a_1a_0 \) and \( y = b_{n-1}b_{n-2} \ldots b_1b_0 \).

I claim that you can express in a succinct Boolean expression whether the carry bit at the \( i \)-th position is 0 or 1.

Let \( c_i \), where \( n \geq i \geq 0 \), denote this bit, i.e., \( c_i = 1 \) if and only if when you do the addition bit by bit, and when you get to the \( i \)-th place, there is a carry bit (from position \( i - 1 \)).

First \( c_0 = 0 \) obviously.

Prove that for \( n \geq i \geq 1 \),

\[
c_i = \bigvee_{i > j \geq 0} \left[ a_j \land b_j \land \bigwedge_{i > k > j} \left[ a_k \lor b_k \right] \right].
\]

If the final answer of the addition of \( x \) and \( y \) is represented by the \( n + 1 \) bit string \( z = d_n d_{n-1} d_{n-2} \ldots d_1 d_0 \), then find and prove a Boolean expression for \( d_i \) in terms of \( a_i, b_i \) and \( c_i \).
Describe a parallel algorithm to compute Addition. The algorithm runs in parallel rounds. In each round, multiple bits can be computed in one batch, and you can use any output bits that have already been computed in previous rounds, including any input bits (round zero). In one step one can do a Boolean AND, OR, NOT operation on these bits. (E.g., one can in two rounds compute $\land_{0 \leq k < n} (a_k \lor b_k).$) The total number of operations should be at most a polynomial (in $n$). The goal for you is to minimize the number of parallel rounds. What is the number of parallel rounds of your algorithm? Prove your answer is correct.

Note:

Please be concise.