CS 372/469: Lab 2, due Mon, 9/24, before lab: 2:19 pm
Total points 100, with extra credit total points: 130

This lab will cover the divide-and-conquer technique we talked about in class. There is no restriction on language for the implementations: use your favorite one.

1. (30 points) Implement the long-hand multiplication algorithm for two n-bit unsigned numbers, with no recursion, no divide-and-conquer, etc. Note that you need to multiply *bitstrings* not integers, and the return value of your program should also be a bitstring. E.g., $x = 11110000$, $y = 10111100$, then if $z = x \times y$, your program should return $z = 1011000001000000$. The truth tables for binary multiplication and binary addition are given in Table 1 and Table 2, for your reference.

2. (30 points) Now implement the divide-and-conquer recursive method we talked about in class, using arity (branching factor) of 3 (ref. textbook, pg. 47, Fig 2.1). You should not assume the number of bits is always even. You will need to take care of this carefully in implementing the algorithm outlined in the book.

3. (30 points) Test the two programs on 5 examples of your choice, please make sure your numbers are reasonably large to see a noticeable difference. Generate plots to compare the running times of the 2 programs with the examples. As in Lab 1, you can use either Gnuplot, or Excel/Google sheets, etc.

4. (*Extra credit 30 points!*) Implement the naïve divide-and-conquer recursive method we talked about in class – the one *without* the Gaussian trick. In other words, the arity of your recursion tree will be 4, instead of 3. You can use the same program you wrote for the previous problem, just that instead of 3, you'll have 4 multiplications. Verify that the running time is the same as the naïve algorithm – $O(n^2)$.

5. (10 points) Write a lab report that describes your work (length max. around 1.5–2 pages)
   (a) Introduction (define the background and the problem)
   (b) Results (show the numbers and figures)

Table 1: Binary multiplication

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Table 2: Binary addition

x	y	z
0	0	0
0	1	1
1	0	1
1	1	10 (carry 1)

(c) Discussions (implications and issues): In particular, talk about examples for which the performance of the algorithms are more or less the same.

How to submit: Upload your **pdf** file on Canvas before the due date. You can use my posted template if you wish, for typesetting your lab, but aren't required to do so.