Your submission *for this assignment* ***must be commented*** *and* ***must include both your name*** *and your* ***student number*** *as a comment at the top of* ***every source file you submit****. Each of your submitted files must use a* ***file name beginning 'comp3007_w23_#########_assignment_06'*** *(replacing the number signs with your own student number) and any* ***submissions that crash*** *(i.e., terminate with an error) on execution will automatically* ***receive a mark of 0****.*

Officially, the Due Date for this Assignment is:

Friday, March 31th, 2023*, at* ***11:59pm EST****.*

Late Submissions are Accepted Without Penalty Until Sunday, April 2nd, by 11:59pm EST.
Submissions received after that will not be accepted and will receive a mark of 0.

The objective of this assignment is allow you to practice with both algebraic data types and tail-call optimization by exploring the creation of three recursive "long division" calculating functions. Long division is actually representative of successive subtractions operations of the divisor (i.e., denominator) from the dividend (i.e., numerator).

Assignment Project Exam Help

For instance, 22 ÷ 7 (for which the quotient is 3 and the remainder is 1) is computed by successive subtractions of 7 from 22 until the difference would be negative; 22 – 7 = 15; 15 – 7 = 8; 8 – 7 = 1, and since three subtraction operations were performed and the final difference was one, the quotient is 3 and the remainder is 1.

https://tutorcs.com

For this assignment:

WeChat: cstutorcs

- you must design and implement a recursive long division function that takes two integer arguments (n.b., the dividend and the divisor, in that order) and produces a 2-tuple of integers (i.e., the quotient and the remainder)
- as the function described above is the *first* of the three long division functions you will write, handle the case of division by zero using Haskell's "error" function (which in turn will require that you consult the Haskell documentation).
- you must design and implement a *second* recursive long division function that takes two integer arguments and is capable of returning both the quotient and the remainder with a single call, but this second version of the function requires that you use a Maybe type to eliminate the need for the "error" that occurs during division by zero. To clarify, this approach will assume that division by zero results in nothing…
- you may wish to write your own selector functions to support the *second* function described above
- you must design and implement a *third* recursive long division function that uses tail-call optimization; this will obviously require that you write a helper function as well, but you can assume that this third function will never be called with a divisor of zero (so you can omit the use of the error function and the use of Maybe)
- you must not use any built-in functions other than "error" and "fst"/"snd" (should you wish to use selector functions).