1. Problem 1 (30 Points)

Using the RPAL string-to-tree-transduction grammar, draw the abstract syntax tree (if possible) for each of the following programs:

(a) x or y -> x | z -> x | y

(b) x eq y -> z -> x| y| x

(c) x aug y, z, 3

(d) x, y, z aug 3

(e) ( (let f(x) = x*2+1) in (f(z) where z=6) )

(f) item (e) above with all the parentheses removed

(g) let x = z and y = 2 * z where z = 3 in x** y**y

(h) let x = y in x * z ** y where z = 4 and p = 3*y

2. Problem 2 (20 Points)

In problem 1, the structure of each program is not apparent because the program is written on a single line as a linear string. Re-write each program on several lines, indenting appropriately to reflect the structure.
3. Problem 3 (25 Points)

The integer operator "**" in RPAL raises an integer to an integer power. Now please implement an RPAL function called "power" to achieve the same functionality. For example (power 3 2) returns the same value as (2**3), which is 8. Your function should use the fact that \( x^{2n} = (x^2)^n \) to reduce the number of multiplications required. You are not allowed to use "**" operator in the function. (Note: Please submit a separate “p3.rpal” with your function for Problem 3. Your "p3.rpal" should be able to pass the RPAL interpreter. It may help a lot if you try your function in a RPAL interpreter before you hand it in)

4. Problem 4 (25 Points)

Implement a function "merge" which takes two lists of numbers in increasing order and merges them into one list of numbers in increasing order. For example,

\[
\text{merge}((1,3,4,7),(2,4,6,7,8)) = (1,2,3,4,4,6,7,7,8).
\]

Test your program in an interpreter (No need to submit a .rpal file)