1. **Problem 1 (18 Points)**
The following RPAL program is overly parenthesized; remove all superfluous parentheses, i.e. minimize the parentheses without changing the meaning of the program.

Also, format the program by writing it on several lines with the appropriate indentation.

```plaintext
let (x =((a * b)-c)) in ((e & f) -> (g -> x ** (x**2)|y)
| (f(x) where (( f(y) = (y*2)) and (x=(2 + (x ** 2)))))
```

**Solution:**
```
let x = a*b-c
in
  e & f -> g -> x**x**2
   | y
   | ( f x
     where ( f y = y*2
               and x = 2+x*2
             )
   )
```

2. **Problem 2 (25 Points)**
Write, test, and debug an RPAL program that computes the "tuple reverse" function:

\[ \text{Rev}(4, 'hello', (3,4), true) = (true, (3,4), 'hello', 4). \]

Please write them on several lines with the appropriate indentation.

**Solution:**
The result may vary. Here is one example:

```
let Rev T = Prev (T,1)
where rec Prev (T,N) = N gr (Order T) -> nil
    | (Prev (T,N+1) aug (T N))
in
Print (4,'hello',(3,4),true)
```

3. **Problem 3 (25 Points)**
Explain the behavior of the following RPAL program, and the type of each identifier which appeared:

Let \( \text{Prod} N = P 1 N \) where rec \( P \) \( \text{Cum} N = N \) eq 1 -> \( \text{Cum} \) | \( P (\text{Cum*N}) \)
Make an honest attempt at figuring it out BEFORE asking the RPAL interpreter.

Solution:
Function Prod takes an arbitrary number of arguments (one at a time) and multiplies them. The multiplication stops when the current argument is 1.
Types:
Prod / P / Print -Function
N(in Prod) / N(in P) / Cum -Integer

4. Problem 4 (32 Points, 4 points each)
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
(a)  \( \rightarrow \)  \( \text{or} \) \( \rightarrow \) \( x \) \( \rightarrow \) \( yzxy \) \\
(b)  \( \rightarrow \)  \( \text{infix} \) \( \rightarrow \) \( x \) \( \text{eq} \) \( yzxy \) \\
(c)  \( \tau \)  \( \rightarrow \) \( \text{aug} \) \( z3 \) \\
(d)  \( \tau \)  \( \text{aug} \) \( z3 \) \\
(e)  \( \text{fcn_form} \) \( \text{where} \) \( \text{G} = \) \\
\( \text{f} \) \( x + \) \( = \) \( 1fzz6 \) \\
\( x2 \) \\
(f)  \( \text{same as (e)} \) \\
(g)  \( \text{let} \) \\
\( \text{and} \) \( ** \) \\
\( xz=yy \) \\
\( 2zz3 \) \\
(h)  \( \text{No AST possible; a definition following 'where' cannot contain 'and' unless the definition is parenthesized.} \)