1. Problem 1 (20 Points)

Give an example of a left associative operator in RPAL and then modify the grammar such that it won't be left or right associative anymore.

**Solution:**
An example is 'or' operator:
B -> Bt ('or' Bt)+
(as done for 'tau' operator.)

2. Problem 2 (20 Points)

According to the semantic definition of RPAL, are RPAL programs meant to be evaluated in PL order or normal order? Why was this choice made?

**Solution:**
Normal order for its better computation time.

3. Problem 3 (20 Points)

Can you give an example of an RPAL program whose root of a fully standardized tree is not lambda or gamma? Can you give an example of an RPAL program in which Ystar is not a leaf in its standardized tree? (If it exists in both cases)

**Solution:**
Answer to the first part: A program with just the number “3”
Answer to the second part: no
4. Problem 4 (20 Points)

Can you write a macro in standard C that `returns' the greatest common divisor of a pair of arguments, without calling a subroutine? Why or why not? [textbook 3rd edition, problem 3.22]

Solution:
No. The macro must either call a subroutine or execute a loop. Not possible in C.

5. Problem 5 (20 Points)

In Section 3.5.4 we noted that while a single min function in C would work for both integer and floating-point numbers, overloading would be more efficient, because it would avoid the cost of type conversions. Give an example in which overloading does not seem advantageous--one in which it makes more sense to have a single function with floating-point parameters, and perform coercion when integers are supplied. [textbook 3rd edition, problem 3.20]

Solution:
When there are many arguments to a function and type conversion happens often inside the body of function. Consider calculating the mean of four numbers (may be integer or may not). To avoid conversion (and therefore not overloading), we should have 16 functions instead of one that calculates the mean of four doubles.