Question 1

Give a regular expression for each of the following languages over \( \Sigma = \{0,1, \ldots, 9\} \).

1. All 5-digit integers that contain no leading zeros.
2. All positive integers that begin with 9 and that are multiples of 5.
3. All strings that begin with 9 and contain three consecutive 1s.

Question 2

Give a DFA for each of the languages in Question 1.

Question 3

Give a context-free grammar for each of the following languages over \( \Sigma = \{a, b\} \).

1. \( a^* b^* \)
2. Strings that contain an same number of a’s as b’s.
3. \( \{a^n b^{n+k} | 0 \leq k\} \)

Question 4

Consider the following grammar.

\[
E \rightarrow E \text{ or } E \mid E \text{ and } E \mid \text{not } E \mid (E) \mid x
\]

1. Prove that the grammar is ambiguous by giving an example sentence for which there are two or more parse trees.
2. Assume that the operators \( \text{or} \) and \( \text{and} \) are left-associative, the operator \( \text{not} \) is right associative, and that the operators have the precedence relation: \( \text{not} > \text{and} > \text{or} \). Rewrite the grammar into one that does not have ambiguity and respects the associativity and the precedence of the operators.

Question 5

Write the following functions in Picat, Haskell, or Python using recursion. No higher-order functions or list comprehensions can be used in the implementations.

1. \( \text{my}_\text{zip}(\text{lst1}, \text{lst2}) \): Let \( \text{lst1} \) be \([A_1, A_2, \ldots, A_n]\), and \( \text{lst2} \) be \([B_1, B_2, \ldots, B_n]\). This function returns the association list \([(A_1, B_1), (A_2, B_2), \ldots, (A_n, B_n)]\).
2. \( \text{lookup}(\text{alist}, x) \): This function returns the value associated with \( x \) in the association list \( \text{alist} \). For example,
   \[
   \text{lookup}([('a', 1), ('b', 2), ('c', 3)], 'b')
   \]
   returns 2.
3. \( \text{replicate} (\text{lst}, n) \): This function replicates the elements of \( \text{lst} \) \( n \) times. For example
   \[
   \text{replicate}(['a', 'b', 'c'], 3)
   \]
   returns ['a', 'a', 'a', 'b', 'b', 'b', 'c', 'c', 'c'].

   1
Question 6

Design a data structure for binary trees, and write the following functions on binary trees in Picat, Haskell, or Python.

1. `leaves(tree)`: This function returns a list of leave values in `tree` from left to right.

2. `deepest(tree)`: This function returns the values in a deepest node in `tree`. If there are multiple such values, then the functions returns the left-most one.

3. `min_max(tree)`: This function returns a pair `(min, max)`, where `min` is the minimum element, and `max` is the maximum element in `tree`. Note that the tree may not be a binary search tree.