Name: ____________________________ RCSID: ________

1. Grammars, ambiguity, precedence and associativity (25 points)

Consider the Boolean expression grammar. *S* and *E* are nonterminals; **`and`, `or`, `not`,** and **`b`** are terminals.

$S \rightarrow E$
$E \rightarrow E$ **and** E | E **or** E | **not** E | **b**

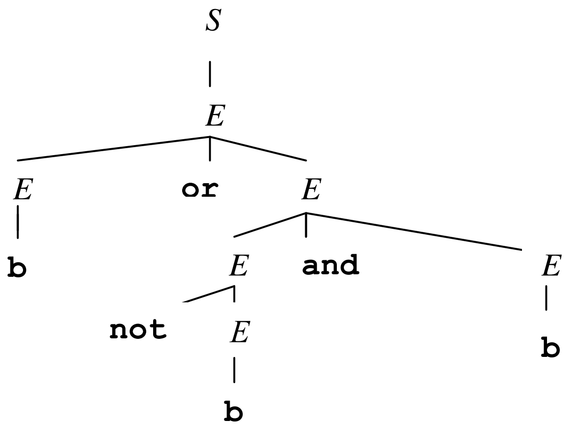
a) **(12 points)** Show that the grammar is ambiguous by drawing all parse trees for expression **`b or not b and b`**.

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Suppose that we wanted **`b or not b and b`** to have only one possible parse tree (call this property P):



b) (5 points) Describe in English the precedence needed to ensure property P.

Assignment Project Exam Help

https://powcoder.com

c) (8 points) Construct an equivalent unambiguous grammar that gives precedence to operators **`or`**, **`and`**, and **`not`** according to property P, and left associativity to **`or`** and **`and`**.

Add WeChat powcoder

2. LL Parsing (22 points)

Consider the following grammar over terminals `0, 1, a,` and `b`.

$$S \rightarrow A\ \texttt{\$\$}$$
$$A \rightarrow B\ A \mid \texttt{0}\ A\ \texttt{1}\ A \mid \varepsilon$$
$$B \rightarrow \texttt{a}\ C\ \texttt{b}$$
$$C \rightarrow B\ C \mid \texttt{0}\ C\ \texttt{1}\ C \mid \varepsilon$$

a) (9 points)* Fill in the table below with the FIRST and FOLLOW sets for the nonterminals:

	FIRST	FOLLOW
A		
B		
C		

b) (10 points)* Fill in the column headings and the entries for the LL(1) parsing table for this grammar: (There may be more columns than needed.)

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

S						
A						
B						
C						

c) (3 points) Is the above grammar LL(1)? Explain briefly why yes, or why not.

3. SLR(1) Parsing (50 points)

We return to the ambiguous grammar for Boolean expressions from Part 1.

$$S \rightarrow E$$
$$E \rightarrow E\ \textbf{and}\ E \mid E\ \textbf{or}\ E \mid \textbf{not}\ E \mid \textbf{b}$$

h) (3 points)* Given the conflict resolution rules from **g)**, what is the maximal number of grammar symbols, terminals or nonterminals, that can appear on the stack?

4. LL and SLR grammars (8 points)

a) (2 points)* Consider this grammar over terminals **`(`**, **`)`**. The grammar is

$$S \rightarrow A$$
$$A \rightarrow AA \mid \mathtt{(}\ A\ \mathtt{)} \mid \varepsilon$$

(1) LL(1) only (2) SLR(1) only (3) LL(1) and SLR(1) (4) neither LL(1) nor SLR(1)

b) (2 points)* Consider this grammar over terminals **`and`**, **`or`**, **`not`**, **`b`**. The grammar is

Assignment Project Exam Help

$$S \rightarrow E$$
$$E \rightarrow \mathtt{and}\ E\ E \mid \mathtt{or}\ E\ E \mid \mathtt{not}\ E \mid \mathtt{b}$$

https://powcoder.com

(1) LL(1) only (2) SLR(1) only (3) LL(1) and SLR(1) (4) neither LL(1) nor SLR(1)

c) (2 points)* Now this one:

Add WeChat powcoder

$$S \rightarrow E$$
$$E \rightarrow E\ \mathtt{and}\ E \mid E\ \mathtt{or}\ E \mid \mathtt{not}\ E \mid \mathtt{b}$$

(1) LL(1) only (2) SLR(1) only (3) LL(1) and SLR(1) (4) neither LL(1) nor SLR(1)

d) (2 points)* Consider the following grammar over terminals **`c`** and **`d`**. The grammar is

$$S \rightarrow A\mathtt{c}A\mathtt{d} \mid B\mathtt{d}B\mathtt{c}$$
$$A \rightarrow \varepsilon$$
$$B \rightarrow \varepsilon$$

(1) LL(1) only (2) SLR(1) only (3) LL(1) and SLR(1) (4) neither LL(1) nor SLR(1)

5. Prolog (20 points)

a) (8 points) Consider the program below. You may assume that the first two arguments are positive integers.

```
d(A, B, 0, A) :- A < B.
d(A, B, Q, R) :- A >= B, A1 is A-B, d(A1, B, Q1, R), Q is Q1+1.
```

Show ALL answers to this query

```
?- d(5, 3, Q, R).
```

`d(A, B, Q, R)` does _________________________

`Q` contains ______________________________

`R` contains ______________________________

b) (12 points)** Write a Prolog predicate `eval` that takes a list representing a boolean expression in **preorder**, and prints its boolean value. `0` and `1` stand for boolean values `false` and `true` respectively.

Assignment Project Exam Help

E.g., `eval([or,0,1],V).` yields `V = 1`, and `eval([and,or,0,and,0,1,1],V).` yields `V = 0`. However, `eval([and,or,0,and,0,1],V).` yields `false.`

https://powcoder.com

Note: You may use built-in or helper predicates as needed. You may use arithmetic operators `+` and `*` to help emulate `or` and `and` respectively.

Add WeChat powcoder