1 Compiler Construction (jdy22)

(a) Describe the inputs and outputs of a lexer and parser. [2 marks]

(b) Here are three grammars for languages with branching, sequencing and variables:

Grammar 1:  
E -> if E then E else E  
E -> E then E end  
E -> id

Grammar 2:  
E -> if E then E else E end  
E -> E then E end  
E -> id

Grammar 3:  
E -> if E then E else E  
E -> do E then E  
E -> id

(i) For each grammar, state whether it is ambiguous, giving an example if it is. [3 marks]

(ii) For each grammar, state whether it is in LL(1), giving a reason if it is not. [3 marks]

(c) Explain the roles of the ACTION and GOTO tables in the LR parsing algorithm, describing the entries and indexes for each table. [4 marks]

(d) Compilers sometimes simplify expressions to make type checking easier or to generate more efficient code. Here are two potential Slang simplifications:

<table>
<thead>
<tr>
<th>expression simplifies to expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>if true then e₁ else e₂ → e₁</td>
</tr>
<tr>
<td>(fun (x:t) -&gt; e₁) e₂ → {e₂/x}e₁ (substitution)</td>
</tr>
</tbody>
</table>

A simplification $e_1 \rightarrow e_2$ is correct for type checking if $e_1$ and $e_2$ have the same type (or are both ill-typed), and correct for optimization if $e_1$ and $e_2$ have equivalent behaviour.

(i) For each simplification, explain under what circumstances it is correct for type checking. [4 marks]

(ii) For each simplification, explain under what circumstances it is correct for optimization. [4 marks]