**CSC 448: Compiler Design: 2015 Spring, Assignment #3**

Last modified 2015 May 15

Student: Yuancheng Zhang

**Purpose:**

To go over theory the theory of LL(1) and LR(1) parsing.

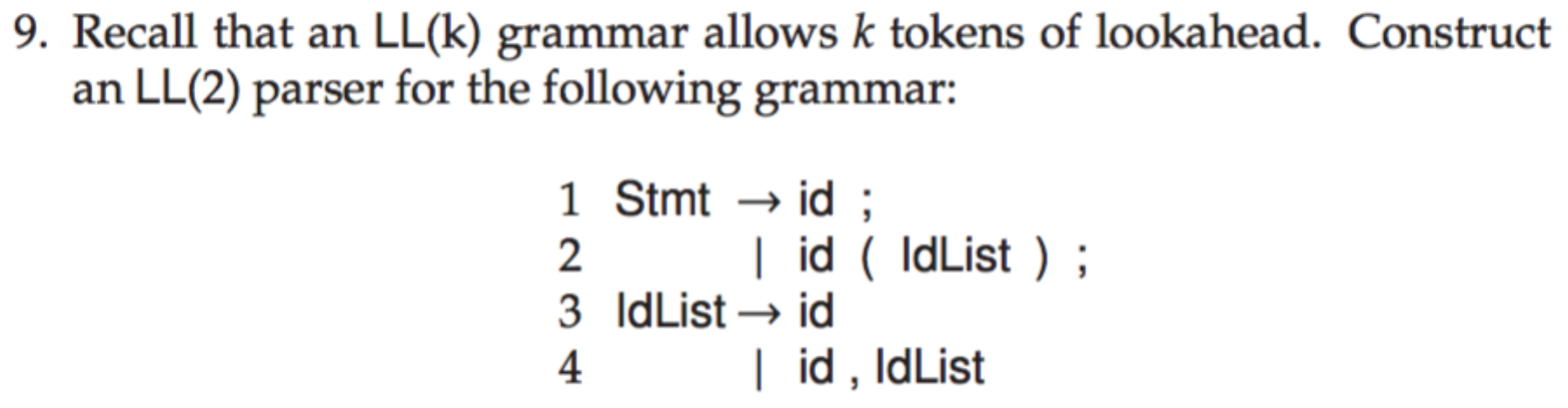
**Computing:**

Please ssh into ctilinux1.cstcis.cti.depaul.edu, or use your own Unix machine.

**Assignment:**

1. **Improving llParserMaker.cpp (40 points)**(empty, see the cpp file)
2. **Questions from the book, chapter 5, pages 173-178 (30 points)**

Question 9



LL(1)

First Set:

terminals:

First( id ) = { id }

First( ; ) = { ; }

First( ( ) = { ( }

First( ) ) = { ) }

non-terminals:

First( Stmt ) = { id }

First( IdList ) = { id }

Follow Set:

Follow( Stmt ) = { $ }

Follow( IdList ) = { ) }

LL(1) Parsing Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | id | ; | ( | ) | $ |
| Stmt | Stmt->id;  Stmt->id(IdList); |  |  |  |  |
| IdList | IdList->id  IdList->Id,IdList |  |  |  |  |

Therefore, this grammer is not LL(1).

LL(2)

First2( Stmt ) -> { id; id( }

First2( IdList ) -> { id id, }

Follow( Stmt ) -> { $ }

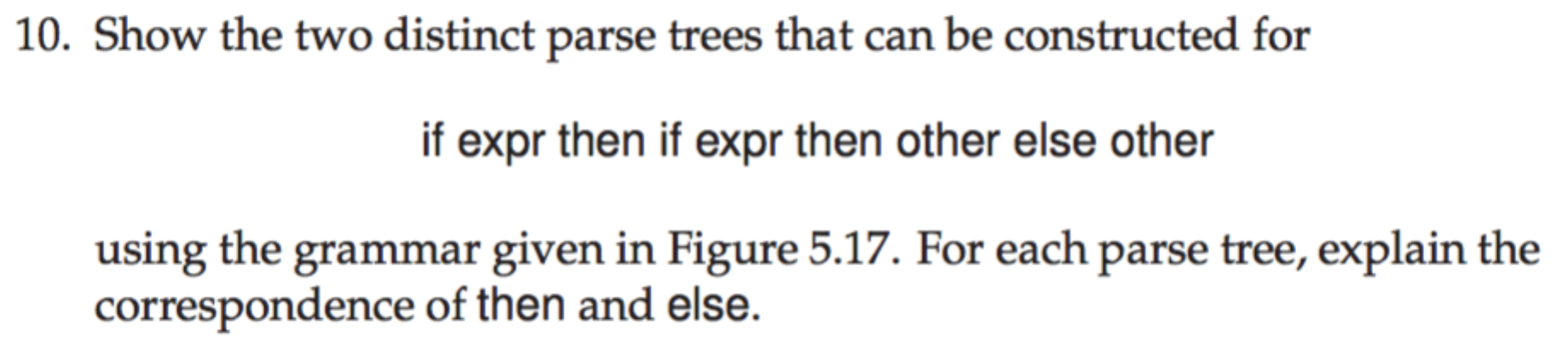
Follow( IdList ) -> { ) }

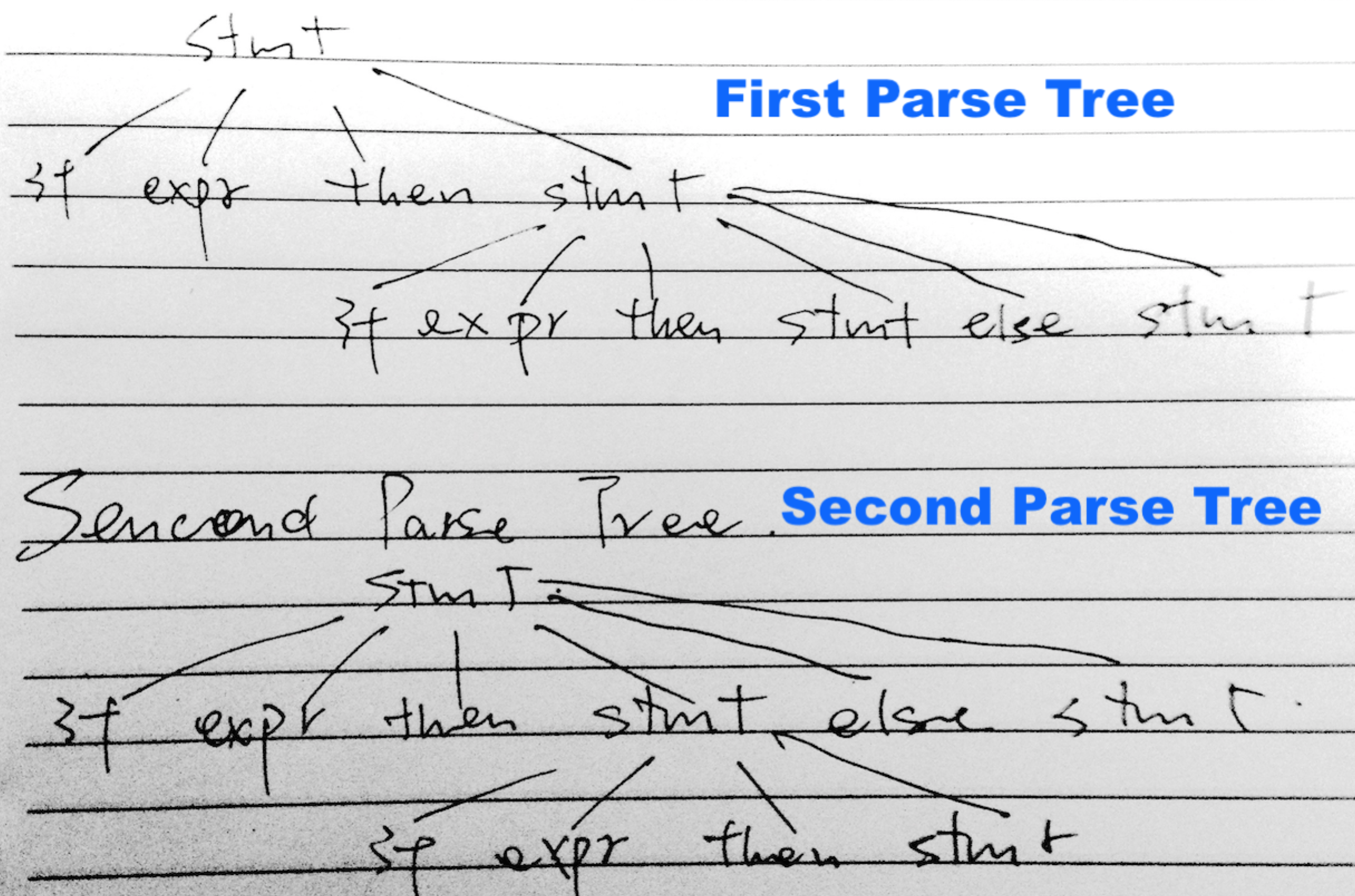
LL(2) Parsing Tabled

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | id$ | id; | id( | id, | *others* |
| Stmt | *error* | Stmt->id; | Stmt->id(IdList); | *error* | *error* |
| IdList | IdList->id | *error* | *error* | IdList->Id,IdList | *error* |

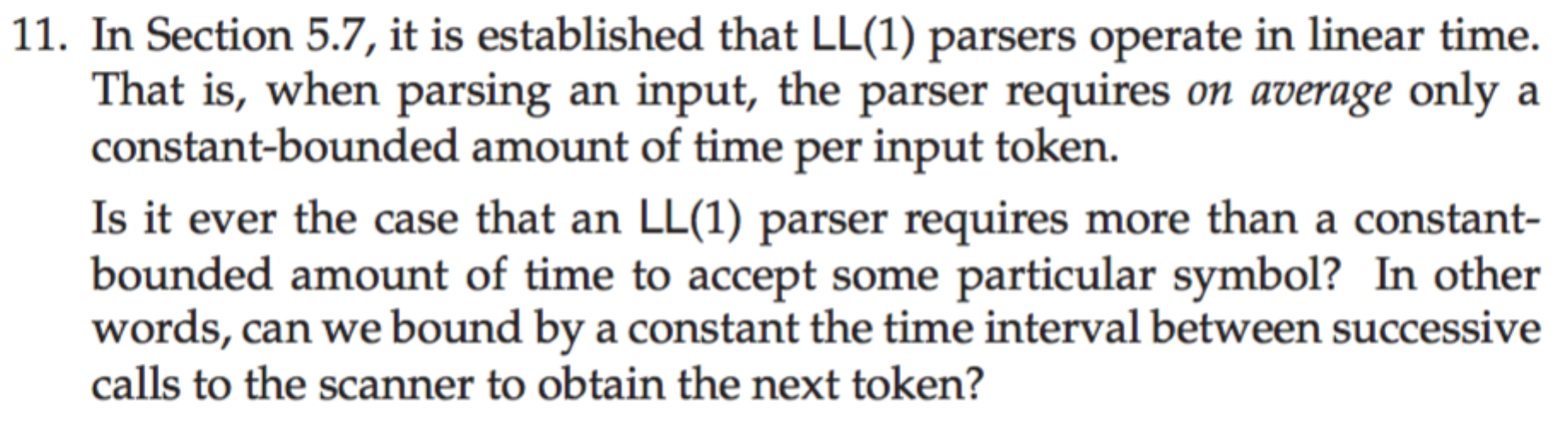
Therefore, this grammar is LL(2).

Question 10





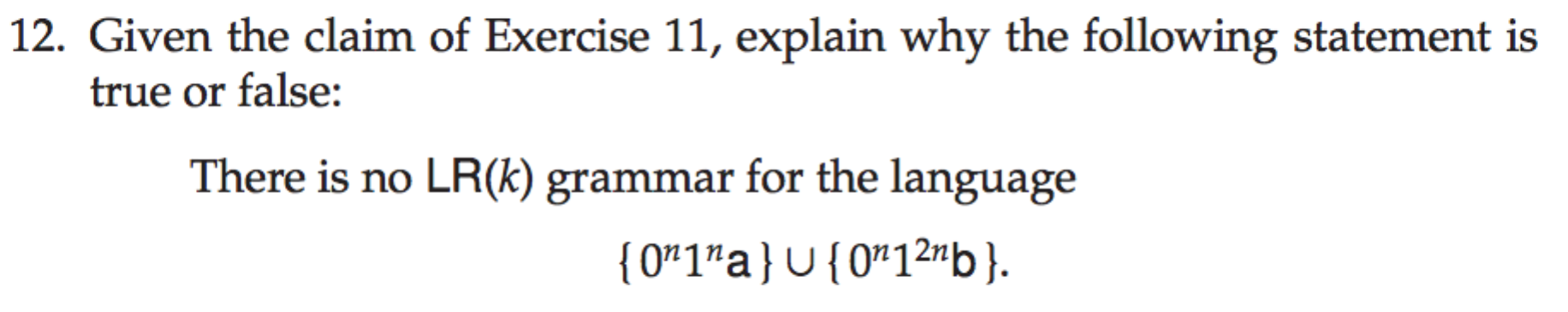
Question 11

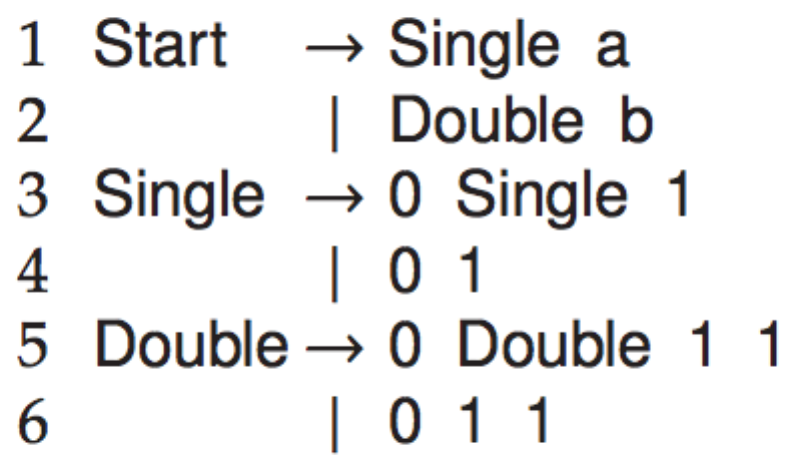


No, we can't. When the parser accepts a token, we don't know how many times the stack will pop or push. If each stack action costs a constant time, we don't know how long it takes before obtain the next token.

1. **Questions from the book, chapter 6, pages 224-233 (30 points)**

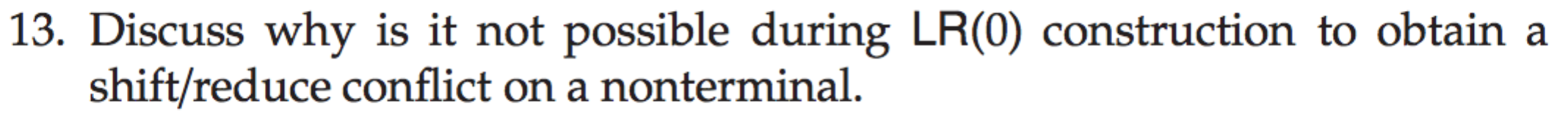
Question 12





It's true. This language is inherently nondeterministic. Only deterministic context-free grammar can be parsed by LR parser. There is no LR(k) grammar for this.

Question 13



If we shift a nontermianl X onto the top of the stack, it means that X is a handle, which is a reduction that also allows further reductions back to the start symbol. X must be reduced definitely. Therefore, there is no shift/reduce conflict on a nonterminal.

Question 40



This grammar is not not LALR(1).

This grammar is ambiguous. Any ambiguous grammar can not be LR.

If **if e then Stmt** is on the stack and **else Stmt** is unprocessed input stream, we don't know whether if e or Stmt is the handle. There is a shift/reduce conflict.