VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
VELLORE ■ CHENNAI
www.vit.ac.in

School of Computer Science and Engineering
Continuous Assessment Test – II, Oct 2017

B. Tech

Course Code : **CSE2002** **Duration: 90mns**

Course Name : **Theory of Computation and Compiler Design** **Max. Marks : 50**

Answer ALL questions [5*10= 50 Marks]

1.

a. The following context-free grammar describes part of the syntax of a simple programming language. Nonterminals are given in capitals and terminals in lower case. VAR represents a variable name and CONST represents a constant. The productions for ASSN-STMT are not given, as they do not affect the question. **[5 Marks]**

PROGRAM → *Procedure STMT-LIST*
STMT-LIST → *STMT STMT-LIST | STMT*
STMT → *do VAR = CONST to CONST { STMT-LIST } | ASSN-STMT*

Show the parse tree for the following:

Procedure
do i = 1 to 100 {
ASSN-STMT
ASSN-STMT
}
ASSN-STMT

b. An RL(0) parser is similar to an LR(0) parser except that it scans the input from right- to left. We can speak of RL(0) items as productions with a dot marking the position of what has been matched so far from the right side instead of the left. For example, if we have an RL(0) item of the form $A \rightarrow x \cdot y$, it means that we have matched y so far and are looking forward to matching x. Similarly, whereas an LR(0) reduce item has the form $A \rightarrow v \cdot$, with the dot at the end, an RL(0) reduce item has the form $A \rightarrow \cdot v$, with a dot at the beginning, since it means we have matched all of v in reverse order. Is there a grammar that is RL(0) but not LR(0)? If so, show what it is and explain both why it is not LR(0) and why it s RL(0). If not, explain why not. **[5 Marks]**

2.

a. Prove that any LR(0) grammar is SLR(1). [3 Marks]

b. Consider the following grammar: [4 Marks]

$X \rightarrow YaYb \mid ZbZa$

$Y \rightarrow \varepsilon$

$Z \rightarrow \varepsilon$

Using the definition of LL(1), explain why the grammar is or is not LL(1).

c. Explain how a LR parser can generate a right-most derivation even though it scans input from left to right [3 Marks]

3. Give pushdown automata that recognize the following language

$L = \{a^* wck \mid w \in \{a, b\}^* \text{ and } k = |w|a \ (k = \text{the number of } a \text{ in } w)\}$

4. Convert the given grammar into CHOMSKY NORMAL FORM (CNF)

$S \rightarrow ASB$

$A \rightarrow aAS|a|\varepsilon$

$B \rightarrow SbS|A|bb$

5.

a. When writing LR parsers, it is common to introduce precedence and associativity declarations to allow the parser to parse ambiguous grammars. However, these declarations cannot be used in LL(1) parsers. By considering the following ambiguous grammar:

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow int$

$E \rightarrow (E)$

Give a proper justification for the above claim? [5 Marks]

b. Explain why this grammar cannot be parsed with an LL(1) parser, even if the parser knew the relative precedences and associativities of addition and multiplication.Consider the following (already-augmented) grammar, which is both LL(1) and LR(1): [5 Marks]

$S \rightarrow A$

$A \rightarrow aBE$

$B \rightarrow bCD$

$C \rightarrow c$

$D \rightarrow d$

$E \rightarrow eFG$

$F \rightarrow f$

$G \rightarrow g$

List which productions are performed and in which order when parsing the string **abcdefg** with an LL(1) parser. Explain why they are performed in this order.