THE OPEN UNIVERSITY OF SRI LANKA
FACULTY OF ENGINEERING TECHNOLOGY
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING
BACHELOR OF SOFTWARE ENGINEERING
ECX6263 – SOFTWARE CONSTRUCTION

Date: 07 December 2016 **Time: 0930 – 1230 hrs.**

Important:

1. This question paper consists of **six** questions.
2. Answer **all** questions in **Part A** (60 marks) and **TWO** questions from **Part B** (40 marks).
3. State your assumptions, if any.

Part A – Answer all questions

The following senariao (reference: *http://www.cs.nmsu.edu/~rth/cs/cs471/f00/calculator.html*) is about a special calculator which has for basic arithmetic operation (ie. Add, Subtract, Divide and Multiplication). You are to design a compiler to implement the following scenario.

A four-function calculator has simple arithmetic 'programs' as shown in the Figure 1. Each 'program' consists of pressing the ON button, followed by any number of calculations followed by pressing the OFF button. The Grammar **G** should only include meaningful programs that produce useful results (any sequence of button pushes is allowed, but not all these sequences produce meaningful results). In particular, notice that parentheses must be matched for a result to be produced. Use the normal notion of parenthesized infix expressions to guide the Grammar **G**. The +/- button changes the sign of the previous operand.

		OFF	ON	
7	8	9	(	+
4	5	6	)	-
1	2	3		×
0	.	+/-	=	÷

Figure 1: Calculator Interface

[Q1] Define the Grammar **G** and write suitable production rules to implement a compiler which is used for implementing the above scenario. (Hint: program → ON OFF | ON calc_list OFF, calc_list → …). [30 Marks]

[Q2] Draw the paser tree for the program ***ON 1 2 3 4 5 6 = OFF*** to demonstrate that your grammar works. [20 Marks]

[Q3] Write LEX implementation syntax for token of the grammar G above (a). [10 Marks]

Part B – Answer ANY TWO questions

[Q4] Consider the following grammar **G**. [*(,)*, ***number*** [0-9]* and ***identifier*** [a-z]* [0-9]* are terminals and all others are non-terminals].

lexp → atom | list
atom → ***number*** | ***identifier***
list → (lexp_seq)
lexp_seq → lexp_seq lexp | lexp

(a) Write leftmost and rightmost derivation for the string ***(var1 77 (var4 var2 var3))***. [04 Marks]

(b) Draw a parse tree for the string of (a). [02 Marks]

(c) Write C like type declarations required to implement an abstract syntax tree structure for the **G**. [05 Marks]

(d) Draw the abstract syntax tree for the given string (a) that would result from the C like type declarations you specified in (c). [03 Marks]

(e) Draw the NDFA and DFA for the ~~given~~ string given in (a). [06 Marks]

[Q5] Consider the grammar rules given below (statement, if-stmt, exp, and else-part are non-terminals and others are terminals).

statement	→ if-stmt | ***other***
if-stmt	→ ***if*** (exp) statement else-part
else-part	→ ***else*** statement | ε
exp	→ ***0|1***

(a) Derive the string: ***if (0) other else other*** [02 Marks]

(b) Define the Chomsky Normal Form (CNF) for CFGs. [04 Marks]

(c) Convert the given grammar into CNF. [12 Marks]

(d) Derive the above string in (a) using new grammar in (c) [02 Marks]

[Q6]

(a) Briefly explain the Top down parsing and Bottom up parsing. [10 Marks]

(b) Briefly explain LR(1) parsing by giving examples. [10 Marks]