00021

THE OPEN UNIVERSITY OF SRI LANKA
FACULTY OF ENGINEERING TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
BACHELOR OF SOFTWARE ENGINNERING
ECX5265 – SOFTWARE CONSTRUCTION

Date: March 22, 2011 **Time: 1400 – 1700 hrs**

Important:

1. This question paper consists of **seven** questions.
2. Answer all questions in **Part A** (60 marks), which is compulsory and **TWO** from **Part B** (40 marks).

Part A

Consider the following description of a compiler to answer Q1 to Q3.

The following grammar is for a compiler, called "*fully parenthesized expressions*" to do some basic arithmetic operations with operands of one digit.

expression ⟶ digit | (expression operator expression)
operator ⟶ + | − | × | /
digit ⟶ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

An arithmetic expression is '*fully parenthesized*' if each operator plus its operands is enclosed in a set of parentheses and no other parentheses occur. As examples, this grammar produces such forms as 3, (5+8) and (9 - ((3+4) ×1)).

Q1

(a) What are the terminals and the non-terminals in this grammar? [2]

(b) Define regular expressions for lexical units (tokens) of this grammar. [4]

(c) Draw a single NFA from your regular expressions. [8]

(d) Define the token table for this compiler. [4]

(e) How do you represent a token in the lexical analyzer of this compiler? Explain with an example. [2]

(f) Describe how the lexical analyzer would process an input string. [5]

(g) Describe the stream of tokens generated by the lexical analyzer for the input string (5+8). [3]

Q2

(a) Write three rules of the syntax analyzer for this compiler. [6]

(b) Describe how the syntax analyzer would process an input (stream of tokens). [6]

(c) Describe the syntax analyzer operation for the input in above Q1(g) [4]

Q3

(a) Explain in briefly, the *context handling* in this compiler. [4]

(b) Define the instructions for the code generation phase of this compiler. Assuming a stack-based (post-fix) system. [6]

(c) By using above instructions in (b), Write the result of the code generation phase when run the expression (4×((3+1)/2)). [6]

Part B

Q4

(a) Draw NFA for the regular expression (ab|b*)*(ba)* over the $\Sigma = \{a, b\}$. [5]

(b) Convert the NFA obtained in (a) to a DFA. [15]

Q5 Consider the following grammar for Boolean expressions (E is a Non terminal)

$$E \longrightarrow E \textit{ or } E \mid E \textit{ and } E$$
$$E \longrightarrow \textit{not } E$$
$$E \longrightarrow (E)$$
$$E \longrightarrow \textit{true} \mid \textit{false}$$
$$E \longrightarrow \textit{ID}$$

(a) Show that this grammar is ambiguous by using the string:

not (ID and ID or ID) and ID [6]

(b) Rewrite the grammar to remove the ambiguity. Make sure that your revised grammar accepts the same language as the original. [10]

(c) Then, derive the above string in (a) using your new grammar. [4]

Q6

(a) Define Chomsky Normal Form for CFGs. [4]

(b) Convert the following grammar to CNF.

$$S \longrightarrow XaX \mid bX \mid Y$$
$$X \longrightarrow XaX \mid XbX \mid \varepsilon$$
$$Y \longrightarrow ab$$ [16]

Q7 A Turing machine (TM) operating over the alphabet $\Sigma = \{0, 1\}$ accepts only the strings of the form $0^n1^n2^n$ $(n > 0)$ and the blank symbol B. For this TM,

(a) Draw the transition graph. [14]

(b) List the moves made for the input '001122' using instantaneous descriptions. [6]

End.