Second Semester EXAM **2024/2025 Academic year** **BACHELOR OF TECHNOLOGY**		
Option : FULL STACK DEVELOPMENT		
Course code/ Title: CSE 408 Compiler design	**Credit value**	4
Course instructor(s)	Mr. FOUTHE WEMBE A.L.	

EXERCISE 1

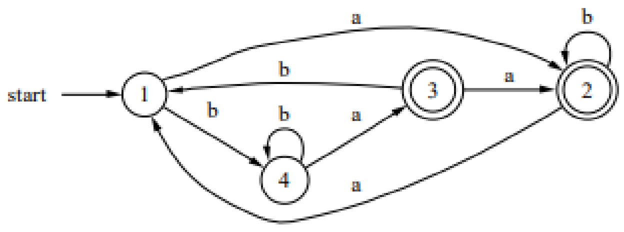
1. For the following expression:
   Position: =initial+ rate*60, Write down the output after each phase of the compiler.
2. Draw the DFA for *a(abb)**
3. Consider the following grammar: $\mathbf{S \to SS + \mid SS * \mid a}$
   a. Show how the string $\boldsymbol{aa + a} *$ can be generated by this grammar
   b. Construct the parse tree of this string
4. Consider the following grammar

   $$
   \begin{aligned}
   S &\rightarrow T * P \\
   T &\rightarrow U \mid T * U \\
   P &\rightarrow Q + P \mid Q \\
   Q &\rightarrow Id \\
   U &\rightarrow Id
   \end{aligned}
   $$

   a. Eliminate Left Recursion
   b. Compute the First and the Follow sets

EXERCISE 2

1. For each of the following regular expressions give two strings that are members of the language it represents and give two that are not:
   a. $a{*}b{*}$
   b. $a(ba){*}a.$
2. Find a regular expression for the language consisting of alternating zeroes and ones.
3. Let M be the following DFA
   a. Can M recognize the string ***aabbab?***
   b. Write down four string accepted by M and the sequence of configurations that shows this.
   c. Write down four strings not accepted by M.



EXERCISE 3

1. Write a context-free grammar for each of the following languages.

   1.1. Sequences of 1 or more numbers separated by + signs. You may use the terminal symbol number to represent a number.
   1.2. Strings over the alphabet **{a, b}** that have the same number of a as b.
   1.3. The language of the regular expression **(xyz)∗(yzx)∗**

2. Consider the following grammar, with the non-terminals **true, false, &&, and||**

   T → **true | false |T && T| T||T**

   2.1. Demonstrate that the grammar is ambiguous by showing at least two parse trees for the string

   true && false ||true.

   2.2. Refactor the grammar so that 1) it is not ambiguous, 2) the && operator is right-associative, 3)

   the || operator is left-associative, and 4) && has higher precedence (binds tighter) than ||

   2.3. Draw the parse tree for **true && false ||true** in your refactored grammar

3. Consider the following grammar, with terminals noun, verb, and modifier:

   ***T* →*SV O*$**

   S* → noun / *MS

   V* → verb /*MV

   O* → ε|*S

   ***M* → modifier**

   3.1. Write the FIRST and FOLLOW sets for this grammar.

   3.2. Fill in the LL(1) parse table for this grammar.

4. Consider the following excerpt of a .lex file:

" "	{ continue; }
"int"	{ return INT; }
"bool"	{ return BOOL; }
"["	{ return LBRACK; }
"]"	{ return RBRACK; }
"->"	{ return ARROW; }
"special:"[a-z]*	{ return SPECIAL; }

   4.1. Write the list of tokens this lexer would produce when given the following string:

   int[] ->special:bool

   4.2. Suppose we wanted to extend the lexer so that if it sees the three-character string end at any

   point in the input, it discards those characters and any input that appears after them, producing

   no further tokens. Write one or more lexer rules that implement this behavior.