**University of Asia Pacific**

**Department of Computer Science and Engineering**

**Mid-Semester Examination Spring-2021**

**Program: BSc in Computer Science and Engineering**

**Course Title:** Compiler Design **Course No.:** CSE 429 **Credit:** 3.00 **Time:** 1.00 Hour. **Full Mark:** 60 **Instruction(s):** Answer any three questions including 1 and 2.

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | a. | Briefly describe:   1. Symbol table and its role 2. Pattern and how we deal with pattern in Syntax analysis. 3. DFA vs NFA | **[ 3**  **+ 3**  **+4**  **=10]** |
|  | b. | Suppose you are dealing with the syntax of C language and you have ∑= [a-zA-Z0-9\_$.]. Here, an identifier can be defined as the following:  **Identifier** can start with a letter or underscore followed by any number of letter, number, underscore or dollar sign.  And, the **floating-point number** can be defined as, numbers that contain floating decimal points. For example, the numbers 5.5, 0.001, and 2345.6789 are floating point numbers. (You can omit the negative floating-point numbers for simplicity)  Construct a DFA (Transition Diagram) for valid   1. **Identifier**, if your ID is even 2. **Floating Number,** if your ID is odd. | **[10]** |
| 2. | a. | Consider the following FA over alphabet ∑= {0, 1, 2}, where **e= epsilon (ε)**    This FA also has two transitions **X** and **Y**.  Where,  **X= Last two digits of your ID%3,** applicablefrom state **C** to state **B**  **Y= 0, if your student ID is odd or Y=2, if your student ID is even,** applicable from state **A** to state **C.**  Now,   1. Draw the transition table of the updated automata. 2. Draw the updated FA diagram. | **[5**  **+5]** |
|  | b. | Using **subset construction method**, convert the above NFA that you got in 2(a) into DFA. | **[10]** |
| 3. | a. | Write regular expression for the following language   1. Set of all strings starts with one letter (**L**) or digit (**D**) and ends with two letters (**L**) followed by one digit (**D**). 2. Set of all strings contains one or more letters (**L**) followed by one Dot (**.**) end with one or more letters (**L**) followed by one digit (**D**). | **[10]** |
|  | b. | Consider the following grammar:  T→ Tx | Ty | T0 | T1| ( S )| x | y | z  T­→ ε  S → S + T | S – T | S \* T | S / T | T   1. Show how the string **“( x + y\*z ) \* x”** can be generated by the grammar by Left Most Derivation and Right Most Derivation. 2. For the given string, state if the grammar is ambiguous or not. | **[10]** |
|  |  | **Or,** |  |
| 4. | a. | Consider the following two CFGs:  i) S🡪 F | De | (S+F)  F🡪 a | DB  B🡪cD | cDSf | Da | ef | ε  D🡪 abc| e| ε  ii) S 🡪A B e | D B  A🡪B | a S| &Bc  B🡪d A S | b | ε  D🡪 A| e B | ε  If **(last two digits of your ID%2==0)** use grammar i) otherwise, use grammar ii). For the predictive parser, generate the first and follow functions from the above grammar. | **[10]** |
|  | b. | Consider the following grammar:  T→ Tx | Ty | T0 | T1| ( S )| x | y | z  T­→ ε  S → S + T | S – T | S \* T | S / T | T   1. Show how the string **“z \* y / ( x0 + x1 )”** can be generated by the grammar by Left Most Derivation and Right Most Derivation. 2. For the given string, state if the grammar is ambiguous or not. | **[3**  **+3**  **+4]** |