library.

Extra

Total Subject - 16

SPRING MID-SEMESTER EXAMINATION-2020

School of Computer Engineering
KIIT Deemed to be University, Bhubaneswar-24

Compiler Design
[CS3002]

Time:1.5 hours **Full Marks:20**

Answer any four questions including Q.No:1 which is compulsory.
The figures in margin indicate full marks. Candidates are required to give answers in their own words as far as practicable and all parts of a question should be answered at one place only.

Q1) Answer all questions. **[1×5]**

(a)How are LL(k) parsers different from LR(k) parsers?

(b) What is input buffering? What is the significance of sentinels in input buffering?

(c)Identify the lexemes and the corresponding tokens in the following statement,

Printf("arr + brr = %d\n", arr>=123);

(d) Consider the grammar

X->X@Y | X#Y | Y
Y->Z$Y | Z
Z->a.

Find the associativity of the operators {@,#,$} and compare their precedence.

(e)Consider the grammar S-> (L) and L-> L,S | a. Derive the string ((a,a),a) using left most derivation and right most derivation.

Q2) Explain all the phases of compiler. Find the output of each phase for the following assignment statement. Assume pos,val and rem are of real type. **[5]**

pos = val * 60 + rem;

Q3) Identifiers in ABC programming language are similar to those in many other languages, but a little different. An identifier: **[5]**

i. Contains any combination of letters, digits, and underscores (for this problem,
assume that letters are only the ASCII characters a-z and A-Z, and digits are 0-9)
ii. Should not begin with a digit.
iii. May optionally have a single **$** or a single @ character at the beginning
iv. May optionally have a single **!** or a single # character at the end.

a. Give a regular definition that generates the set of valid ABC identifiers.
b. Draw a transition diagram that accepts the set of valid ABC identifiers.

Q4) Consider the following grammar G: **[5]**

E->E+T | T
T ->id | id[] | id[X]
X ->E,E | E

a. Eliminate left recursion in G to construct G1 with L(G1)=L(G).
b. Perform left factoring for G1 to construct G2 with L(G2)=L(G1).
c. Compute the FIRST sets for all non-terminals in G2.
d. Compute the FOLLOW sets for all non-terminals in G2.
e. Construct a LL(1) parsing table for the grammar G2.

Q5) Design a recursive descent parser in for parsing the string generated by the grammar, **[5]**

A -> x C | z
B -> y A
C -> B x | A y B