Name: Enrollment No:	UPES

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
End Semester Examination, December 2021

Course: Compiler Design **Semester: V**
Program: B.Tech- CSE (All Branches) **Time: 03 Hrs**
Course Code: CSEG 3015 **Max. Marks: 100**

Instructions: Attempt all the questions.

SECTION A

1. **Each Question carries 4 Marks.**
2. **Instruction: Write short notes / Select the correct answer(s).**

S.No.		CO
Q 1	Describe the process of bootstrapping through an example of your own choice using T-diagram.	**CO1**
Q 2	Draw the structure of regular definitions. Moreover, describe the advantage of using them with a simple example.	**CO2**
Q 3	Discuss the contents of the activation record.	**CO4**
Q 4	Discuss the different error recovery techniques in the syntax analysis phase.	**CO4**
Q 5	Define flow graph. What are the advantages of constructing a flow graph.	**CO5**

SECTION B

1. **Each question carries 10 marks.**
2. **Instruction: Write short/brief notes.**

Q 6	Write short notes on the following: (a) Cross Compiler vs. Native Compiler (b) Single Pass Compiler vs. Multi Pass Compiler (c) Regular Grammar vs. Finite Automata **(d)** Front End Compiler vs. Back End Compiler	**CO1**
Q 7	Write lex programs for the following: (a) To count the number of +ve and -ve real numbers. (b) To count the number of ‘scanf’ and ‘printf’ statements in a C program. In addition, replace them with ‘readf’ and ‘writef’ statements respectively.	**CO2**
Q 8	Consider the following sets of LR(1) items in the states of a LR(1) parser:	**CO3**

State 0:
$[A \rightarrow \bullet a, b]$
$[A \rightarrow a \bullet, c]$
$[B \rightarrow a \bullet, b]$

State 2:
$[A \rightarrow \bullet a, c]$
$[A \rightarrow a \bullet, b]$
$[B \rightarrow a \bullet, a]$

State 1:
$[A \rightarrow \bullet a, a]$
$[A \rightarrow \bullet a, b]$
$[B \rightarrow a \bullet, b]$

State 3:
$[A \rightarrow \bullet a, b]$
$[B \rightarrow \bullet a, b]$

a) Find all shift/reduce and reduce/reduce conflicts. List the LR(1) items and lookaheads causing conflicts.
b) What states would be merged in a LALR(1) parser?
c) Are any new reduce/reduce conflicts introduced in the LALR(1) parser?
d) Explain why LALR(1) parsers will not introduce new shift/reduce conflicts.
e) What is the relationship between LALR(1) and SLR(1) parsers?

Q 9 — **CO4**

(a) Demonstrate the difference between L-attributed SDD and a non L-attributed SDD with the suitable example.

(b) Translate the following expression into triple representation:
x[i] = interest(p, n, r) + y[i] +p

OR

Production	Semantic Rules
$T \rightarrow F\ T'$	$T'.inh = F.val$ $T.val = T'.syn$
$T' \rightarrow *F\ T'_1$	$T'_1.inh = T'.inh \times F.val$ $T'.syn = T'_1.syn$
$T' \rightarrow \varepsilon$	$T'.syn = T'.inh$
$F \rightarrow$ **digit**	$F.val =$ **digit**$.lexval$

For the above mentioned SDD, draw the annotated parse tree for " 3 * 9" and list the possible evaluation order(s).

SECTION C

1. **Each question carries 20 marks.**
2. **Instruction: Write long answer.**

Q10 — **CO3**

(a) Consider the following grammar: -

A → AcB | cC | C

B → bB | id

C → CaB | BbB | B

Construct the first and follow sets for the grammar. Also design a LL(1) parsing table for the grammar.

(b) Construct the operator precedence parsing table by computing leading and trailing for every non-terminal in the following grammar:

.

$$S \rightarrow N$$

$$N \rightarrow V = E \,\#$$

$$N \rightarrow E$$

$$E \rightarrow V$$

$$V \rightarrow id$$

$$V \rightarrow * E$$

Q 11

For the following problems, consider this code:

```
<S1>            a := 1
<S2>            b := 2
<S3>     L1:    c := a + b
<S4>            d := c - a
<S5>            if (...) goto L3
<S6>     L2:    d := b * d
<S7>            if (...) goto L3
<S8>            d := a + b
<S9>            e := e + 1
<S10>           goto L2
<S11>    L3:    b := a + b
<S12>           e := c - a
<S13>           if (...) goto L1
<S14>           a := b * d
<S15>           b := a - d
```

(a) What are the basic blocks? What is the control flow graph?

(b) Depth-first order selects nodes in the order they are visited (start by visiting the root node) and then recursively visiting every child of each node (if the child has not been visited before). Note that the order in which children are visited is random. What are all the possible results of depth-first traversal on the control flow graph?

CO5

(c) Using depth-first order, is it possible to visit a child before its parent? For which depth-first ordering(s) of the control flow graph does this occur?

(d) Postorder selects nodes (starting from root) after visiting every child of that node (if the child has not been visited before). Note that the order in which children are visited is random. What are all the possible results of Postorder traversal for the control flow graph? Reverse Postorder simply reverses the node ordering found by a Postorder traversal of the graph. What are the possible Reverse Postorder traversals of the control flow graph?

OR

(a) Construct the DAG for the following basic block: (5 Marks)

$s = q * r$
$t = p + q$
$q = q * r$
$p = t - s$

	(b) What is the principle of working of peephole optimization? (5 Marks) (c) Consider the following source code in a high level language and convert it into intermediate codes followed by a machine code. Assume that all the data types are integers and take 4 bytes. (10 Marks) *a = b + c * d;* *a = x/(y+z) - b * (c+d);* *a = x[i] + 1;* *x[i] = y[x[i]];* *x[i][j] = y[i][j] + z[k][j];*		