Qn. Set Code-7 (SOT)

Semester: 6th
Programme: B.Tech
Branch: CSE

SPRING END SEMESTER EXAMINATION-2024

6th Semester B.Tech

COMPILER DESIGN

CS 3008

(For 2021 & Previous Admitted Batches)

Time: 2 Hours 30 Minutes   Full Marks: 50

Answer any FIVE questions.
Question paper consists of two SECTIONS i.e. A and B.
Section A is compulsory.
Attempt any Four question from Sections B.
The figures in the margin indicate full marks.
Candidates are required to give their answers in their own words as far as practicable and all parts of a question should be answered at one place only.

SECTION-A

1. Answer the following questions: [1 × 10]

(a) Find the **FIRST** of all the non-terminals in the following grammar:

$$S \rightarrow AaAb \mid BbBa$$

$$A \rightarrow \varepsilon$$

$$B \rightarrow \varepsilon$$

(b) Differentiate between Synthesized and Inherited attribute in Syntax-Directed Definition (SDD).

(c) Find the viable prefixes of the right sentential form of "aaAb" using the following grammar:

$$S \rightarrow AA$$

$$A \rightarrow aA \mid b$$

(d) Which phase of the compiler detects the error in the following C code:

```
#include <stdio.h>
void main() {
int3num= 1234;
}
```

(e) What are the positives of non-inclusion of lexical analyzer in parser?

(f) Consider the following SDT:

P$_1$: $A \rightarrow B + C\{A.val = B.val + C.val\}$

P$_2$:$A \rightarrow B \cdot C\{A.val = B.val * C.val,\ B.val = C.val\ \}$

Select the correct option:

i) P$_1$ is L-attributed but P$_2$ is not L-attributed

ii) P$_1$ is S-attributed but P$_2$ is L-attributed

iii) Both P$_1$ and P$_2$ are S-attributed

iv) None of the above

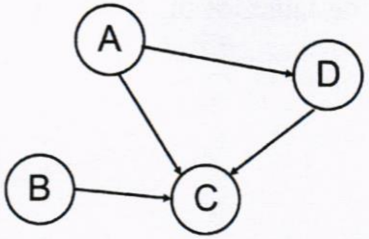
(g) Consider the grammar $G = \{(S, S', E), (i, t, e, a, b), S, P\}$ with the following production rules:

$$P = \{S \rightarrow iEtSS' \big| \, a, S' \rightarrow eS \mid \varepsilon, E \rightarrow b\}$$

Find the entry $M[S', e]$ in the LL(1) parsing table.

(h) Differentiate between machine dependent and machine independent code optimization on the basis of their objective.

(i) Considering the following dependency graph, write any one possible order in which we can evaluate the attributes at the various nodes of a parse tree.



(j) Write down the optimized version of the 3-address code, assuming usual associativity and precedence rules:

$$s = a \wedge (b + c) * b + c + e$$

SECTION-B

2. (a) A compiler uses a two-buffering scheme to read the input. Provide a pseudo code that will scan the input through the buffers in an efficient way. Make use of sentinels. [5]

   (b) Write the Regular Definition for unsigned numbers (integer or floating point) in C language. Also construct the State Transition Diagram from the above definition. [5]

3. (a) Construct the LL(1) parser for the following grammar: [5]

$$\boldsymbol{S \rightarrow (L) \mid a}$$

$$\boldsymbol{L \rightarrow L\,,S \mid S}$$

   Parse the given input string ***(a,a)*** using the above specified grammar.

   (b) Construct the SLR(1) parsing table for the following grammar: [5]

$$\boldsymbol{S \rightarrow AS \mid b}$$

$$\boldsymbol{A \rightarrow SA \mid a}$$

   Check whether the given grammar is SLR(1) or not. Give justification to your answer.

4. (a) Construct the LALR(1) parsing table for the following grammar: [5]

$$E \rightarrow E + T \mid T$$

$$T \rightarrow F \uparrow T \mid F$$

$$F \rightarrow id$$

(b) Construct the annotated parse tree and evaluate the value of the string ***-110*** using the following SDT: [5]

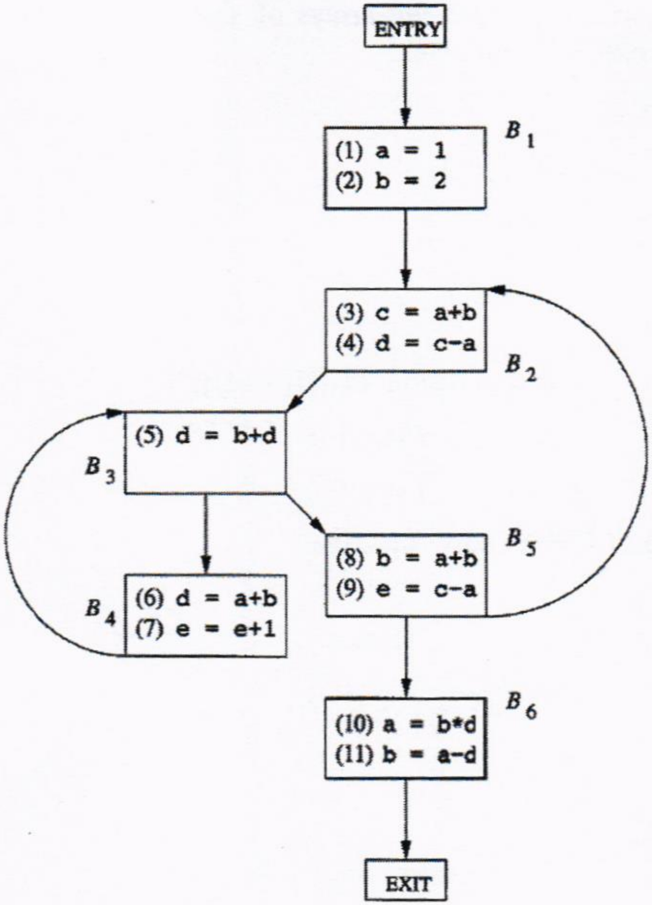
Production	Attribute Rule
$number \rightarrow sign\ list$	$list.pos = 0$ $if\ sign.neg$: $number.val = -list.val$ $else$: $number.val = list.val$
$sign \rightarrow +$	$sign.neg = false$
$sign \rightarrow -$	$sign.neg = true$
$list \rightarrow bit$	$bit.pos = list.pos$ $list.val = bit.val$
$list_0 \rightarrow list_1\ \ bit$	$list_1.pos = list_0.pos + 1$ $bit.pos = list_0.pos$ $list_0.val = list_1.val + bit.val$
$bit \rightarrow 0$	$bit.val = 0$
$bit \rightarrow 1$	$bit.val = 2^{bit.pos}$

5. (a) What is a DAG? Why DAG is used in the process of compiler design? Construct the DAG for the following statement: [5]

$$((a + b) - ((a + b) * (a - b))) + ((a + b) * (a - b))$$

(b) Translate the intermediate code corresponding to the given expression $-(\boldsymbol{a} * \boldsymbol{b}) + (\boldsymbol{c} + \boldsymbol{d}) - (\boldsymbol{a} + \boldsymbol{b} + \boldsymbol{c} + \boldsymbol{d})$ in quadruple, triple and indirect triple form. [5]

6. (a) For the following flow graph: [5]



i) Identify the loops of the flow graph.

ii) Statements (1) and (2) in $B_1$are both copy statements, in which a and b are given constant values. For which uses of a and b can we perform copy propagation and replace these uses of variables by uses of a constant? Do so, wherever possible.

iii) Identify any global common subexpressions for each loop. Eliminate these subexpressions if possible.

(b) Generate the three-address code for the following C code, where a and b are arrays of size 20×20, and word size is of four bytes. [5]

```
add = 0;
i = 0;
j = 1;
do
{
            add = add + a[i][j] * b[j][i]
                   i = i + 1
                   j = j + 1
}while (i <= 20 && i <= 20);
```

*****