COMPILER DESIGN
(CSEN 4111)

Time Allotted : 3 hrs **Full Marks : 70**

Figures out of the right margin indicate full marks.

Candidates are required to answer Group A and <u>any 5 (five)</u> from Group B to E, taking <u>at least one</u> from each group.

Candidates are required to give answer in their own words as far as practicable.

Group – A
(Multiple Choice Type Questions)

1. Choose the correct alternative for the following: **10 × 1 = 10**

(i) A bottom up parser generates __________
(a) Right most derivation (b) Rightmost derivation in reverse
(c) Leftmost derivation (d) Leftmost derivation in reverse.

(ii) Assume S1 = the number of states in SLR(1) parser
S2 = the number of states in LALR(1) parser
S3 = the number of states in CLR(1) parser
Then which of the following is true?
(a) S1=S2=S3 (b) S1=S2 <S3
(c) S1<S2=S3 (d) S1<S2<S3.

(iii) Given below are the regular expressions:
(a|b)* (ii) (a*b*)* (iii) (ab)*
Which of them are equivalent:
(a) (i) and (ii) only (b) (i) and (iii) only
(c) (ii) and (iii) only (d) None of them.

(iv) Find the number of tokens in the following code:
if(x>=y) z=0;
(a) 8 (b) 9 (c) 10 (d) 11.

(v) Reduction in strength means
(a) Replacing run time computation by compile time computation
(b) Removing loop invariant computation
(c) Removing common sub expression
(d) Replacing a costly operation by a relatively cheaper one.

(vi) Which of the following phase of compiler is an optional phase?
(a) Semantic analysis (b) Lexical analysis
(c) Intermediate code generation (d) Code optimization.

(vii) If a grammar is LALR(1) then it is necessarily:
(a) SLR(1) (b) LR(1) (c) LL(1) (d) None of the options.

(viii) Which of the following is not a peephole optimization?
(a) Removal of a dead code
(b) Elimination of multiple jumps
(c) Elimination of loop invariant computations
(d) None of the above options.

(ix) Which of the following is not the content of the activation record?
(a) Temporary variables (b) Return values
(c) Local variables (d) Address descriptor

(x) An S-attributed grammar can be evaluated ____________
(a) Top down (b) Bottom up
(c) Both (d) None.

Group – B

2. (a) Explain the different phases of a compiler, showing the output of each phase, using the example of the following statement:

```
for (i=0;i<10;i++)
    a=a+10;
```

[(CO4) (Remember/LOCQ)]

(b) Construct DFA directly from **(not by generating NFA)** of the following regular expression:
L= (a|b)* ab [(CO2)(CO1) (Understand/LOCQ)]

6 + 6 = 12

3. (a) What language does the regular expression (0|1)*0(0|1)(0|1) generate? [(CO2)(Remember/LOCQ)]
(b) Consider the context free grammar:
S -> **SS+ | SS* | a**
Show how the string **aa+a*** can be generated by this grammar and construct a parse tree for this string. [(CO3) (Understand/LOCQ)]
(c) What is the front end and back end of a compiler? [(CO4) (Analyse/IOCQ)]

3 + (3 + 3) + 3 = 12

Group – C

4. (a) $S \rightarrow 1AB \mid \in$ {∈ represents NULL}
$A \rightarrow 1AC \mid 0C$
$B \rightarrow 0S$
$C \rightarrow 1$
Test whether the given grammar is LL(1) or not
[(CO3)(Remember/LOCQ)(Analyse/IOCQ)(Apply/IOCQ)]
(b) Is it true that LR(k + 1) is more powerful than the LR (k)? Justify your answer.
[(CO2, CO3)(Evaluate/HOCQ)]

(c) "A grammar containing left recursion cannot be LL(1), similarly a grammar containing right recursion cannot be LR(1)." Is this a correct statement? Explain clearly. [(CO2, CO3)(Evaluate/HOCQ)]

6 + 3 + 3 = 12

5\. (a) Consider the grammar:

S -> AS | b
A -> SA | a

(i) Construct the SLR parsing table for this.
(ii) Show all moves of parsing for the input: abab
[(CO4) (Understand/LOCQ)]

(b) Eliminate left recursion from the following grammar:

S -> Ab|b
A -> AC|Sd|€ [(CO2) (Understand/LOCQ)]

(4 + 4) + 4 = 12

Group – D

6\. (a) Consider the SDT shown below:

E → TE1
E1 → +T {print("+");} E1 | ∈
T →num {print("num.lex.val");}

Now modify this SDT in such a way so that an LR parser can carry out the translation on an input "9 + 5 + 2". [(CO4)(Remember/LOCQ)]

(b) (i) Generate the three address code for the code snippet given below:

```
i = 2 * n + k;
while(i)
    {
      i = i – k;
    }
```

(ii) Then implement the three address code in triples, and indirect-triples.
[(CO3, CO4)(Remember/LOCQ)(Analyse/IOCQ)]

3 + (3 + 3 + 3) = 12

7\. (a) Divide the following code into basic blocks and draw the flow graph.

1. f=1;
2. i=2;
3. if (i>=x) goto (8)
4. f=f*i;
5. t1 = i+1;
6. i=t1;
7. goto (3)
8. goto calling program. [(CO1) (Remember/LOCQ)]

(b) Consider the following SDT. If an LR parser carries out the translations on an input string "aabbccdb", what is the output?

S -> aaS {print("a");}
| bA {print("b");}

```
        | b      {print("c");}
A -> bcA         {print("d");}
        | cdS    {print("e");}
```

[(CO3) (Understand/LOCQ)]

(c) What is the significance of a flow graph? [(CO1)(Analyze/IOCQ)]

(3 + 2) + 5 + 2 = 12

Group – E

8. (a) Explain the terms with example:
(i) Register descriptor (ii) Address descriptor. [(CO4) (Remember/LOCQ)]

(b) Translate the following code into machine code and show the register and address descriptors while the instructions are generated. Assume that two registers are available.
(i) x = y + z (ii) w = p + y (iii) y = y +z (iv) p = w - x
[(CO3) (Understand/LOCQ)]

4 + 8 = 12

9. (a) Can you optimize the given C code segment (*do not forget to mention the code optimization technique that you have used*):

```
i = 0;
j = 0;
while(i< n && j < n)
{
        x += 4 * j + a[i];
        y -= a[i] – 4 * j;
        i++;
        j++;
}
```

[(CO5, CO6)(Analyse/IOCQ)(Evaluate/HOCQ)]

(b) Design the flow graph for the code snippet given below:

```
int fib(unsigned int n)
{
int f0 = 0, f1 = 1, f2, i;
    if(n <= 1)
        return n;
    else
    {
        for(i = 2; i<= n; i++)
        {
            f2 = f0 + f1;
            f0 = f1;
            f1 = f2;
        }
        return f2;
    }
}
```

[(CO6)(Understand/LOCQ)(Analyse/IOCQ)]

6 + 6 = 12

Cognition Level	LOCQ	IOCQ	HOCQ
Percentage distribution	70%	19%	11%

Course Outcome (CO):

After the completion of the course students will be able to

CO1: This course will enable a student to understand the major phases of compilation including the front- and backend. They are expected to have an overview of how a real life compiler works.

CO2: After completion of this course, the students are expected to develop knowledge of Lex and YAAC tools.

CO3: The students should be able to understand various necessary tasks related to compiler construction, like token identification, grammar writing, type conversion, and storage management.

CO4: Students will learn to generate intermediate codes and actual machine codes targeting a particular architecture.

CO5: Students should acquire a detailed idea regarding optimization of generated code across various phases of the compilation process.

CO6: After completion of this course, students should be able to apply various optimization techniques for dataflow analysis.

*LOCQ: Lower Order Cognitive Question; IOCQ: Intermediate Order Cognitive Question; HOCQ: Higher Order Cognitive Question

Department & Section	**Submission link:**
CSE - A	https://classroom.google.com/w/NDA1MjE1ODUzNjMy/tc/NDYzODM0OTA0MTY0
CSE - B	https://classroom.google.com/c/NDA1MzUzNzI5MzYw/a/NDYzODU1NDI4ODA2/details
CSE - C	https://classroom.google.com/w/MTIyNDM4MjA5OTg2/tc/NDYzODMxMjk1NjMw