# CS61

| USN | 1 | M | S | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|

RAMAIAH
Institute of Technology

(Autonomous Institute, Affiliated to VTU)
(Approved by AICTE, New Delhi & Govt. of Karnataka)
Accredited by NBA & NAAC with 'A' Grade

## EXAMINATIONS SEPTEMBER /OCTOBER 2020
## SUPPLEMENTARY SEMESTER / GRADE IMPROVEMENT/ RE –REGISTERED CANDIDATES

| | | | | | |
|---|---|---|---|---|---|
| **Program** | : | **B.E. : Computer Science and Engineering** | **Semester** | : | **VI** |
| **Course Name** | : | **Compiler Design** | **Max. Marks** | : | **100** |
| **Course Code** | : | **CS61** | **Duration** | : | **3 Hrs** |

**Instructions to the Candidates:**

- Answer any one full question from each unit.



### UNIT – I

| | | | | |
|---|---|---|---|---|
| 1. | a) | Explain why the analysis portion of a compiler is separated into lexical analysis and parsing. Describe the tasks of a lexical analyzer in the design of a compiler. | CO1 | (06) |
| | b) | i. Consider the alphabet Σ={a,b}. Define a short, possibly shortest, regular expression that generates strings over Σ that contain exactly one "a" and atleast one "b".<br>ii. Give regular expressions for Email ID validation.(Hint: abc@yahoo.com) | CO1 | (06) |
| | c) | Discuss about the different strategies that a parser can follow to recover from a syntactic error. | CO1 | (08) |
| 2. | a) | Explain the input buffering technique using sentinels with example. | CO1 | (06) |
| | b) | Describe the front end of the compiler phases with the output representation of each phase. | CO1 | (08) |
| | c) | Describe why left factoring is required for implementing top down parser?<br>Left factor the given grammar.<br>S→abCg \| abA<br>A→cdC<br>C→cdef \| cd | CO1 | (06) |

### UNIT – II

| | | | | |
|---|---|---|---|---|
| 3. | a) | Consider the following Context Free Grammar<br>S→XS \| dS \|ε<br>X→Y \|Zb \|aY<br>Y→ cZ<br>Z→e<br>i. Compute the FIRST and FOLLOW sets for each non-terminal symbol.<br>ii. Construct the parsing Table for a non-recursive predictive parser for the grammar.<br>iii. Is the Grammar LL (1)? Justify.<br>iv. Show how the predictive parser parses the string 'dace' using the stack. | CO2 | (14) |
| | b) | Demonstrate that the following grammar is not SLR. Justify why the grammar can't be in LR (0).<br>E→1E1 \| 1 | CO2 | (06) |
| 4. | a) | Design the LR (1) parser for the given grammar. Also explain the procedure for the computation of Closure of the set of items.<br>S→hBe<br>B→A \| ε<br>A→x \| t | CO2 | (09) |

b) Explain the need of Augmentation in LR grammars and how the given grammar can be changed to an augmented grammar? CO2 (06)

c) Compute the FOLLOW for all the Nonterminals in the given grammar CO2 (05)

i. S→SA |0 |ε
A→aS1

ii. A→B C
B→Ax| x
C→yC |y| ε

**UNIT – III**

5. a) Design an S-Attributed Definition to compute S.val, the decimal number value of an input binary string. For example, the translation of the string 101.11 should be the decimal number 5.75. CO3 (08)
The given Grammar generates binary numbers with a decimal point.
S→ L. L | L
L→L B | B
B→ 0|1

b) Write SDD for typesetting boxes. Modify the given CFG to include superscripts denoted by operator sup between boxes. If box B2 is a superscript of box B1, then the position the baseline of B2 is 0.6 times the point size of B1 above the baseline of B1. CO3 (08)
Add the new production and rules to the SDD.
S→B
B→$B_1$ $B_2$
B→$B_1$ sub $B_2$
B→text

c) Explain how an L-attributed SDD can be translated into an SDT with a suitable example. CO3 (04)

6. a) Write S- Attributed Definition for computing the number of bits in the given binary value.(Hint: If the input is a binary value 11101, from the root value N.count = 5) CO3 (05)
G:
N→D
D→$D_1$ B | B
B→0 | 1

b) Design a Syntax Directed Definition for simple type declarations. Give the Annotated Parse Tree and Dependency graph for the expression: int a, b, c. CO3 (09)
Grammar:
D→T L
T→int
L→ L,id |id

c) Consider the Fibonacci function given CO3 (06)

```
int fib(int n){
   int t,s;
   if(n==0 || n==1) return 1;
   s=fib(n-1);
   t=fib(n-2);
   return s+t;
}
```

i. Which function creates the space on the stack for each element?
ii. What does the stack and its activation records look like the first time
fib (1) is about to return?
iii. Generate an Activation Record for the Fibonacci function where n=3.

**UNIT – IV**

| | | | | |
|---|---|---|---|---|
| 7. | a) | Translate the control flow statement into three address code using fall through technique. Draw Annotated Parse Tree for the statement given.<br>if (a==b && b==c \|\| c==d) x=1; | CO4 | (05) |
| | b) | Generate three address code for the following statements.<br>i. n=f(a[i])<br>ii. x=f(y+1)+2<br>iii. g=gcd(x-y,x) | CO4 | (09) |
| | c) | Explain how the type conversions are translated into three address code using the function 'widen'? | CO4 | (06) |
| 8. | a) | Translate the switch statement into three address code<br>switch (5-4){<br>case 0: a=0;<br>case 1: a=9;<br>}<br>} | CO4 | (05) |
| | b) | Explain the unification algorithm of a pair of nodes in a type graph. | CO4 | (05) |
| | c) | Construct DAG and generate its corresponding three address code for the following expression.<br>**a=b+-c + (b+-c)**<br>Draw the Syntax Tree and generate its equivalent three address code for the same expression.<br>Does both generates the same three address code sequence? Justify. | CO4 | (10) |

**UNIT – V**

| | | | | |
|---|---|---|---|---|
| 9. | a) | Discuss about the primary tasks encountered in the design of a code generator. | CO5 | (07) |
| | b) | Generate code for the following three-address statements assuming a and b are arrays whose elements are 4-byte values.<br>i. if x < y goto L<br>ii. a[j] = c<br>iii. x = y + z | CO5 | (09) |
| | c) | Determine the costs of the following instruction sequences:<br>LD R0, y<br>LD R1, z<br>ADD R0, R0, R1<br>ST x, R0 | CO5 | (04) |
| 10. | a) | Consider the program fragment:<br>int fib(int num)<br>{<br>if(num == 0 \|\| num == 1)<br>return num;<br>else<br>return fib(num-1) + fib(num-2);<br>}<br>i. Translate the program into three-address statements.<br>ii. Construct the Basic Blocks and flow graph.<br>iii. Identify the loops in your flow graph. | CO5 | (10) |

b) Generate code for the following three-address statements assuming stack allocation where register SP points to the top of the stack. CO5 (06)

```
call p
call q
return
call r
return
return
```

c) Discuss how the liveness and next-use information is determined for each statement in a basic block. CO5 (04)

**********