RAMAIAH
INSTITUTE OF TECHNOLOGY

USN	1	M	S							

(Autonomous Institute, Affiliated to VTU)
Bangalore – 560 054

SEMESTER END EXAMINATIONS – MAY/JUNE 2017

Course & Branch	**:**	**B.E. – Computer Science & Engineering**	**Semester**	**:**	**VI**
Subject	**:**	**Compiler Design**	**Max. Marks**	**:**	**100**
Subject Code	**:**	**CS1561/CS612**	**Duration**	**:**	**3 Hrs**

Instructions to the Candidates:
- Answer one full question from each unit.

UNIT - I

1.	a)	Explain the phases of the compiler and show the translation of each phase for the assignment statement. Assume all variables are integers. Math_form=$a^2+2ab+b^2$	CO1	(10)
	b)	Construct transition diagram for recognizing unsigned numbers. Sketch the transition diagram to implement it.	CO1	(04)
	c)	Consider the Grammar G: S□ SS+ | SS* | A | ε A□ Sa | a Apply the algorithm to eliminate left recursion from the grammar.	CO2	(06)
2.	a)	Describe why Left factoring is required for Top Down Parsing. Do Left Factoring for i. W□his | her | he |me ii. V□is | was | wait | water| at iii. S□fCs | wCs | dwC | iEts | iEtsEs | iEtsEsEs E□b C□c	CO2	(09)
	b)	Explain how to specify tokens with suitable examples	CO1	(05)
	c)	Discuss about the front end of compiler and how symbol table helps in developing the front end.	CO1	(06)

UNIT – II

3.	a)	Show that the following grammar S□ A a | bAc | dc | bda A□d is LALR(1)	CO2	(14)
	b)	Describe the viable prefixes with example.	CO2	(06)
4.	a)	Construct predictive parser for the following grammar. rexpr□rexpr + rterm | rterm	CO2	(14)

rterm□rtermrfactor | rfactor
rfactor□rfactor * | rprimary
rprimary□a|b

b) Write algorithm to compute canonical collection of sets of LR(1) items. CO2 (06)

UNIT - III

5. a) Distinguish between the terms CO3 (10)
   i) Inherited attribute and Synthesized attribute
   ii) Annotated Parse Tree and attribute grammar
   iii) S-Attributed Definition and L- Attributed Definition
   iv) Dependency Graph
   v) Activation tree and Activation record

   b) Design an S-attributed SDD to compute S.val (Each grammar separated with semicolon.) CO3 (10)

G: S→ L && L; S→ L || L; S→ !L; L→0 |1

For example, the translation of string "1&&1||!1" should be the decimal value 1.
Give an Annotated Parse Tree for the input **"1&&1||!1".**

6. a) Generate the SDT for typesetting boxes for the following grammar (Each grammar separated with semicolon.) CO3 (08)
   S□B; B□BB | B sub B | text

   b) Illustrate how the desk calculator is implemented on a bottom-up parsing stack with the help of semantic actions. CO3 (08)

   c) Draw the dependency graph for the following expressions CO3 (04)
   i) int a, b, c
   ii) char a, b, c, d

UNIT – IV

7. a) Translate the following expression into three address code, quadruple and triple. CO4 (09)
   (Assume all are integer declaration and width =4)
   i) f=fib(n-1,n-2)+1
   ii) a*(b+ -c)+(b+ -c)
   iii) a=c+b[i]+d[j]

   b) Give the semantic actions for the translation of expression to three address code. CO4 (05)

   c) Construct DAG for the following expressions CO4 (06)
   i. a+a+(a+a+a+(a+a+a+a))
   ii. a+b+a+b
   iii. ((x+y)*(x-y))+(x+y)-(x*y)

8. a) Differentiate type synthesis and type inference. Explain how to introduce type conversions into expression evaluation with actions embedded. CO4 (04)

   b) Generate the semantic rules for the following productions using fall-through technique. (Each grammar separated with semicolon.) CO4 (06)
   B□E rel E

B□ B || B
E□ id

c) Construct the Annotated Parse Tree for the expression using backpatching x<100 || x> 200 && x!=y
Show the steps in backpatch process for generating intermediated code. CO4 (10)

UNIT - V

9. a) Discuss in detail the main tasks for the code generator. CO5 (06)
b) Differentiate between Address Descriptor and Register Descriptor CO5 (04)
c) Write Three address code and construct the basic blocks for the following program segment CO5 (10)

```
rev=0;
while(num>=0){
  dig= num%10;
  rev=rev*10 +dig;

}
```

10. a) Explain the code generation algorithm and generate assembly language code for the following expression. CO5 (10)

X= a+b+c

b) Convert the following program fragment into **three address code** and obtain CO5 (10)

i) Basic Blocks
ii) Flow graph
iii) Identify the loops if any

Code:

```
a=0;
b=2;
if(a<b){
a=1;
}
else { b=1;}
```

****