Roll No.: _

Amrita Vishwa Vidyapeetham

B.Tech. Second Assessment Examinations – March 2018

Sixth Semester

Computer Science and Engineering

15CSE311 Compiler Design

Time : Two hours | Maximum : 50 Marks

Answer all questions

PART-A (5 * 3 =15 Marks)

1. Consider the following possible sets of LR(1) items in the states of a shift/reduce parser:

State 0:	State 1:
[A → a • , b]	[A → b • , a]
[B → a • b, b]	[B → b • b , b]
[C → b a •, a]	[C → a b • , a]

a. For the items in State 0, list any conflicts that exist and describe for what lookaheads they occur.

b. For the items in State 1, list any conflicts that exist and describe for what lookaheads they occur.

2. Given the following grammar

[0]	S' → S
[1]	S →XX
[2]	X → aX
[3]	X →b

Perform LR(1) item reduction on the following canonical collection of LR(1) items

I_0:	S' -> •S, $ S -> •XX, $ X -> •aX, a/b X -> •b, a/b	I_4:	X -> b•, a/b
I_1:	S' -> S•, $	I_5:	S -> XX•, $
I_2:	S -> X•X, $ X -> •aX, $ X -> •b, $	I_6:	X -> a•X, $ X -> •aX, $ X -> •b, $
I_3:	X -> a•X, a/b X -> •aX, a/b X -> •b, a/b	I_7:	X -> b•, $
		I_8:	X -> aX•, a/b
		I_9:	X -> aX•, $

3. Given the ACTION/GOTO table above, show the parse with the current stack contents are **4 A 2** (i.e., current state is 2) and the remaining input is **"bb$".**

Note*: start the stack with 4 A 2 and input with "bb$"*

State	ACTION			GOTO	
	b	c	$	S	A
0	Shift 2	Reduce $S \rightarrow Bb$	Reduce $S \rightarrow cB$	1	3
1	Shift 4	Reduce $S \rightarrow c$	Reduce $S \rightarrow c$	3	2
2	Shift 3	Shift 4	Accept	2	4
3	Reduce $A \rightarrow Ab$	Reduce $A \rightarrow b$	Reduce $A \rightarrow Ab$	2	1
4	Reduce $A \rightarrow \varepsilon$	Shift 3	Accept	4	2

4. Consider the following C program

```
[1] void f( int k )
[2] {
[3]         int k = 0;
[4]         while (k)
[5]         {
[6]             int k = 1;
[7]         }
[8] }
[9] void main()
[10]        {
[11]                int  _ _a, b, c;
[12]                char* s;
[13]                _ _a +b=c;
[14]                printf("%s", c);
[15]                for(int i=0; i<b; i++)
[16]                {
[17]                    if (s=="i")
[18]                        s=20;
[19]                    c=c/i;
[20]                    f(s,c);
[21]                }
[22]                printf("%d",i);
[23]        }
```

Specify the line numbers with following semantic errors:

i. Type mismatch.
ii. Reserved identifier misuse.
iii. Multiple declaration of variable in a scope.
iv. Accessing an out of scope variable.
v. Actual and formal parameter mismatch.
vi. Division by 0.
vii. Dereference of NULL pointer

5. Say whether each attribute of a non-terminal is inherited or synthesized and why. Show the value of the attributes of G after parsing ¬(A ∧ (A ⇒ B)). *Note: need not draw parse tree or dependence graph, use maximum 4 sentences to answer the question*.

Production	Semantic rules
$G \rightarrow F$	$G.q = F.q$
	$F.b = true$
$F \rightarrow F_1 \wedge F_2$	$F.q = F.b\ ?\ And(F_1.q, F_2.q) : Or(F_1.q, F_2.q)$
	$F_1.b = F.b$
	$F_2.b = F.b$
$F \rightarrow F_1 \vee F_2$	$F.q = F.b\ ?\ Or(F_1.q, F_2.q) : And(F_1.q, F_2.q)$
	$F_1.b = F.b$
	$F_2.b = F.b$
$F \rightarrow \neg F_1$	$F.q = F_1.q$
	$F_1.b = \neg F.b$
$F \rightarrow F_1 \Rightarrow F_2$	$F.q = F.b\ ?\ Or(F_1.q, F_2.q) : And(F_1.q, F_2.q)$
	$F_1.b = \neg F.b$
	$F_2.b = F.b$
$F \rightarrow (F_1)$	$F.q = F_1.q$
	$F_1.b = F.b$
$F \rightarrow id$	$F.q = F.b\ ?\ id.lexeme : Neg(id.lexeme)$

6. Bottom parser:
    a. Complete the formula in terms of bottom up parsing stack action,
       |parse tree nodes| = __________________
    b. Express the total number steps to perform shift/reduce action to validate a given string in terms of its parse tree nodes.
    c. Derivation and reduction are related. Give the relationship.
    d. The partially completed parse tree of a shift reduce parser is called as

       ____
    e. A handle is given as pair < ____________, ____________>. Give its meaning.
    f. Give the general format of a sentential form. Mark the left context, hotspot/ critical spot, lookahead and unvisited portion.
    g. Give the general form of an complete and partially complete LR(1) item.

7. Consider the following (already augmented) grammars for the language a*:

$$S \rightarrow A$$
$$A \rightarrow Aa$$
$$A \rightarrow \varepsilon$$

   and

$$S \rightarrow A$$
$$A \rightarrow aA$$
$$A \rightarrow \varepsilon$$

    a. Construct the canonical collection of LR(1) items for both the grammars separately.
    b. Construct the LR(1) Parse table for both the grammars separately.
    c. Trace the stack for an input string "aa$" using both the LR(1) parse table. However, the LR(1) parser for one of these grammars will use O(n) space in its parsing stack when run on the string , while the other parser will only use O(1) stack space. Identify which grammar's parser uses O(n) stack space and which grammar's parser uses O(1) stack space.

8. Consider the following grammar productions for evaluating positive constant expressions. Assume you have an attribute E.zero that is set to TRUE if the expression is zero, and FALSE if it is non-zero (or undefined). Also assume that the terminal ***CONST*** has an attribute ***val*** representing its value.

E → CONST	E.zero = (CONST.val = =0)
E → E1 + E2	E.zero = ??
E → E1 * E2	E.zero = ??
E → E1 / E2	E.zero = ??
E → (E1)	E.zero = ??

    a. Add rules to the attribute grammar to calculate E.zero for each grammar production.
       Hint: *+ and * can be written using logical operators. Divide by zero is undefined. Fill the ??*

b. Provide the annotated parse tree for the expression (1 + 0) / (2 * 0) and show the calculation for E.zero for each node. Mark the dependence graph and perform topological sort.

9. Consider the following CFG with generates the language $L(G)=\{a^m b^n c^p : \text{where } m, n,p>= 1\}$

$S \rightarrow A\ B\ C$
$A \rightarrow a\ A$
$A \rightarrow a$
$B \rightarrow b\ B$
$B \rightarrow b$
$C \rightarrow c\ C$
$C \rightarrow c$

a. Define an attribute grammar to generate $L(G)=\{ a^n b^n c^n : \text{where } n>= 1\}$. The attribute *equal* takes the value *T* and *F* for true and false respectively. All the non terminals have only synthesised attributes: Attribute(S) = {equal}, Attribute({A,B,C}) = {count}

b. Draw the annotated parse tree for the input string "aabbcc". Show the dependence graph and mark the right order of evaluation.

10.

a. What is dynamic dependence-based method?

b. Consider the following grammar perform a dynamic dependence method to evaluate the attributes of the input string: 110.101

[1]	$N \rightarrow L\,.\,R$	N.val = L.val +R.val
[2]	$L \rightarrow B$	L.val = B.val L.length =1
[3]	$L_1 \rightarrow BL_2$	L_1 .length = L_2 .length +1 L_1 .val = B.val * $2^{L2\ .length}$ + L_2 .val
[4]	$R \rightarrow B$	R.val = B.val
[5]	$R_1 \rightarrow BR_2$	R_1 .val = (B.val + R_2.val) /2
[6]	$B \rightarrow 0$	B.val=0
[7]	$B \rightarrow 1$	B.val=1

Note: show the annotations with colour differentiation.

c. Why is it a nice property to have an attribute grammar that has only synthesized attributes?

d. Why is a circular attribute grammar a bad property?