# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, RAMAPURAM CAMPUS
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## ANSWER KEY SUBMISSION

| | | | |
|---|---|---|---|
| **Date of Exam & Session** | **29/03/2023 & AN** | **Category of Exam** | **CLA2** |
| **Course Name** | **Compiler Design** | **Course Code** | **18CSC304J** |
| **Name of the Faculty submitting** | **Ms. Jayalakshmi<br>Ms.P.Vidyasri** | **Date of submission of Answer Key** | **31/03/2022** |
| **Department to which the faculty belongs to** | **CSE** | **Total Marks** | **50** |

**PART - A (10x1 = 10)**
**ANSWER ALL THE QUESTIONS**

| Q.No | Questions | Marks |
|---|---|---|
| 1 | Which of the following function is called the canonical collection of LR(0) item?<br>a) FIRST( )<br>b) GOTO( )<br>c) COMPUTE( )<br>d) FOLLOW( )<br>Answer: b | 1 |
| 2 | Identify which of the following tree is the pictorial identification of the derivation?<br>a) The oct tree<br>b) The parse tree<br>c) The binary tree<br>d) The derivation tree<br>Answer: b | 1 |
| 3 | Identify which of the following derivations does a top-down parser use while parsing an input string?<br>a) Leftmost derivation<br>b) Leftmost derivation in reverse<br>c) Rightmost derivation<br>d) Rightmost derivation in reverse<br>Answer: a | 1 |
| 4 | What is the TRAILING(S) for the following grammar?<br>S→S-B\|B<br>B→B*A\|A<br>A→(S)\|id<br>a)TRAILING(S)={-,*,),id}<br>b)TRAILING(S)={-,*,(,)}<br>c)TRAILING(S)={-,*,(,id}<br>TRAILING(S)={-,*,(}<br>Answer: a | 1 |
| 5 | Reverse of a right most derivation is called ----------.<br>a)reduction<br>b)production<br>c)handle<br>d)base<br>Answer: c | 1 |
| 6 | Which of the following derivations does a top-down parser use while parsing an input string?<br>a) Leftmost derivation<br>b) Leftmost derivation in reverse<br>c) Rightmost derivation<br>d) Rightmost derivation in reverse<br>Answer: a | 1 |



**Signature of the Faculty**

| | | |
|---|---|---|
| 7 | Which one of the following is a top-down parser?<br>a) Recursive descent parser<br>b) Operator precedence parser<br>c) An LR(k) parser<br>d) An LALR(k) parser<br>Answer: a | 1 |
| 8 | Identify why the grammar A→AA\|(A)\|∈ is not suitable for predictive-parsing?<br>a) Ambiguous<br>b) Left recursive<br>c) Right recursive<br>a) d) An operator grammar<br>Answer:b | 1 |
| 9 | LEADING(S) for the following grammar?<br>S→S-B\|B<br>B→B*A\|A<br>A→(S)\|id<br>a)LEADING(S)={-,*,),id}<br>b)LEADING(S)={-,*,(,)}<br>c)LEADING(S)={-,*,(,id}<br>d)LEADING(S)={-,*,(}<br>Answer: c | 1 |
| 10 | b) which of the following grammar rules violate the requirements of an operator grammar? P, Q, R are nonterminal and s, r, s, t are terminals.<br>1.P→QR<br>2. P→QsR<br>3. P→ε<br>4. P→QtRr<br>a)1 only<br>b)1 and 3 only<br>c)3 and 4 only<br>d)2 and 4 only<br>Answer: b | 1 |

**PART - B (4x4= 16)**

**ANSWER ALL THE QUESTIONS**

| Q.No | Questions | Marks |
|---|---|---|
| 11 | Define a context free grammar.<br>Answer:<br>Context free grammar is a formal grammar which is used to generate all possible strings in a given formal language. Context free grammar G can be defined by four tuples as:<br>G= (V, T, P, S) where,<br>G describes the grammar<br>T describes a finite set of terminal symbols.<br>Non-terminal -> (V U T)* | 4 |
| 12 | Identify the grammar by eliminating Left Recursion:<br>E → E + T \| T<br>T → T * F \| F<br>F → (E) \| id<br>Answer:<br>Comparing E → E + T \| T with A → A α \| β<br>A = E, α = +T, β = T<br>A → A α \| β is changed to<br>A → βA′ and<br>A′ → α A′ \| ε<br>A → βA′ means E → TE′<br>A′ → α A′ \| ε means E′ → +TE′ \| ε | 4 |
| 13 | Perform Shift Reduce Parsing for the following<br>S->(L)\|a L->L,S\|S for the input string: (a,(a,a)) .<br>Stack \| Input Buffer \| Parsing Action | 4 |

| | | | |
|---|---|---|---|
| | $ | (a,(a,a))$ | Shift |
| | $( | a,(a,a))$ | Shift |
| | $(a | ,(a,a))$ | Reduce S → a |
| | $(S | ,(a,a))$ | Reduce L → S |
| | $(L | ,(a,a))$ | Shift |
| | $(L, | (a,a))$ | Shift |
| | $(L,( | a,a))$ | Shift |
| | $(L,(a | ,a))$ | Reduce S → a |
| | $(L,(S | ,a))$ | Reduce L → S |
| | $(L,(L | ,a))$ | Shift |
| | $(L,(L, | a))$ | Shift |
| | $(L,(L,a | ))$ | Reduce S → a |
| | $(L,(L,S | ))$ | Reduce L →L, S |
| | $(L,(L | ))$ | Shift |
| | | | |
| | $(L,(L) | )$ | Reduce S → (L) |
| | $(L,S | )$ | Reduce L → L, S |
| | $(L | )$ | Shift |
| | $(L) | $ | Reduce S → (L) |
| | $S | $ | Accept |

14 Distinguish between Top-down and Bottom-up parser. — 4

Answer:

| Top-down parser | Bottom-up parser |
|---|---|
| Parse tree can be built from root to leaves. | Parse tree can be built from leaves to root. |
| This is simple to implement. | This is complex to implement. |
| It is applicable to small class of languages. | It is applicable to a broad class of languages. |
| Various parsing techniques are:<br>• Recursive descent parser<br>• Predictive parser | Various parsing techniques are:<br>• Shift reduce parser<br>• Operator precedence parser<br>• LR parser |

15 Enumerate the concepts of Operator Precedence parser with an example. — 4

Operator Precedence parsing is a type of shift-reduce parsing that uses a set of rules to determine the order of operations in an expression. The following are the concepts of Operator Precedence parsing:

Operator Precedence: Each operator in the grammar is assigned a precedence level, which determines its priority in the order of operations. Operators with higher precedence levels are evaluated first, and operators with equal precedence levels are evaluated based on their associativity.

Shift Operation
Reduce Operation
Conflict Resolution
Error Handling

16 Elaborate the computation rules of FOLLOW — 4

Follow (A) is defined as the collection of terminal symbols that occur directly to the right of A.

FOLLOW(A) = {a|S ⇒* αAaβ where α, β can be any strings}

Rules to find FOLLOW

1. If S is the start symbol, FOLLOW (S) ={$}

2. If production is of form A → α B β, β ≠ ε.

(a) If FIRST (β) does not contain ε then, FOLLOW (B) = {FIRST (β)}

Or

(b) If FIRST (β) contains ε (i. e. , β ⇒* ε), then

FOLLOW (B) = FIRST (β) − {ε} ∪ FOLLOW (A)

∵ when β derives ε, then terminal after A will follow B.

3. If production is of form A → αB, then Follow (B) ={FOLLOW (A)}.

**PART - C (2x12 = 24)**

**ANSWER THE QUESTIONS**

| Q.No | Questions | Marks |
|---|---|---|
| 17(a) | Define Recursive Descent Parser. And Construct Recursive Descent Parser for the following grammar:<br>E →i E'<br>E'→ + i E' \| ε<br>Answer:<br>Recursive Descent Parser is a technique which may or may not require backtracking process.It uses procedures for every non-terminal entity to parse strings.It is a type of top-down parsing built from a set of mutually recursive procedures where each procedure implements one of non-terminal of grammar.It contains several small functions one for each non- terminals in grammar and accepts all kinds of grammars.<br><br>int main ()<br>{E(); // E is a start symbol.<br>if (l == '$') // if l == $, it represents the end of the string and "l" is lookahead.<br>Printf("Parsing Successful");<br>}E()// Definition of E, as per the given production<br>{if (l == 'i')<br>{match('i');<br>E'();}}<br>E'()// Definition of E' as per the given production<br>{if (l == '+') {<br>match('+');<br>match('i');<br>E'();} //The second condition of E'<br>else<br>return ();}<br>match (char t)// Match function<br>{if (l == t)<br>{l = getchar(); } //Lookahead points to next token<br>else<br>printf("Error"); } | 4<br><br>8 |
| 17(b) | Construct a Predictive Parsing Table for the following grammar:<br>E → TE′<br>E′ → +TE′ \| ε<br>T → FT′<br>T′ → FT′ \| ε<br>F → (E) \| id<br>Answer:<br>Step 1: Elimination of Left Recursion & perform Left Factoring<br>There is no left recursion and left factoring in Grammar.<br><br>Step 2: Computation of FIRST<br>FIRST(E) = FIRST(T) = FIRST(F) = {(, id}<br>FIRST (E′) = {+, ε}<br>FIRST (T′) = {*, ε}<br><br>Step 3: Computation of FOLLOW<br>FOLLOW (E) = FOLLOW(E′) = {), $}<br>FOLLOW (T) = FOLLOW(T′) = {+, ), $}<br>FOLLOW (F) = {+, *, ), $}<br>Step 4: Construction of Predictive Parsing Table<br>Create the table, i.e., write all non-terminals row-wise & all terminal column-wise. | 1<br><br>3<br><br>4 |

| | id | + | * | ( | ) | $ |
|---|---|---|---|---|---|---|
| E | E → TE′ | | | E → TE′ | | |
| E′ | | E′ → +TE′ | | | E′ → ε | E′ → ε |
| T | T → FT′ | | | T → FT′ | | |
| T′ | | T′ → ε | T′ →* FT′ | | T′ → ε | T′ → ε |

4

| | | | | | | |
|---|---|---|---|---|---|---|
| F | F → id | | | F → (E) | | |

Construct the parsing table for SLR parser for the following
S->L=R
S-> R
L->*R
L->id
R->L
Show the parsing action for "id=id"

Step 1: Compute the LR(0) items for the grammar The LR(0) items for the given grammar are:
S' -> .S
S -> .L = R
S -> .R
L -> .* R
L-> .id
R -> .L

I0: S' –> •S
S –> •L = R
S –> •R
L –> •*R
L –> •id
R –> •L

I1: S' –> S•

I2: S –> L• = R
R –> L•

I3: S –> R•

18(a)

I4: L –> *•R R –> •L
L –> •*R
L –> •id I5: L –> id•

I6: S –> L =•R
R –> •L
L –> •*R
L –> •id

I7: L –> *R•

I8: R –> L•

I9: S –> L = R•

Step 2: Compute the FOLLOW sets for each non-terminal The FOLLOW sets for the given grammar are:
FOLLOW(S) = {$, =}
FOLLOW(L) = {$, =, *}
FOLLOW(R) = {$, =, *}

Step 3: Construct the SLR parsing table

6

| State | id | * | = | L | R |
|---|---|---|---|---|---|
| 0 | s4 | | | 2 | 3 |
| 1 | | | | | |
| 2 | s4 | | | 2 | 6 |
| 3 | r2 | s5 | | | 7 |
| 4 | r4 | r4 | s8 | | |
| 5 | r5 | r5 | | | |
| 6 | r3 | s5 | | | 7 |
| 7 | r1 | r1 | | | |
| 8 | | | | | 9 |
| 9 | s4 | | | 2 | 6 |

4

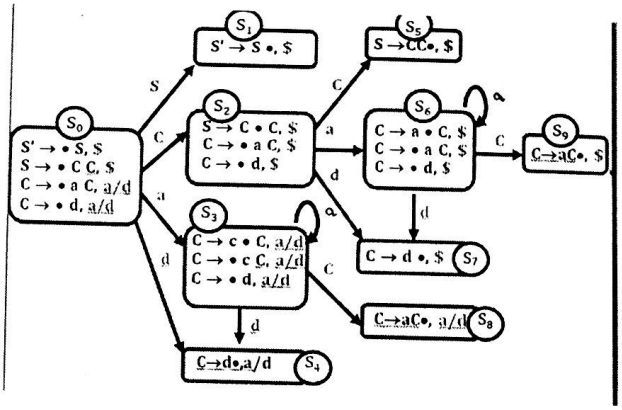
Input String id=id:
0 id=id$ Shift 4
04 id=id$ Reduce L -> id
03 id=id$ Shift 5
035 id=id$ Reduce R -> L
037 id=id$ Reduce S -> R
036 id=id$ Shift 8
0368 id=id$ Shift 4
03684 id=id$ Reduce L -> id
03683 id=id$ Shift 5
036835 id=id$ Reduce R -> L
036837 id=id$ Reduce S -> L = R
036 id=id$ Shift 4
0364 id=id$Top of Form

2

**18(b)**

Consider the grammar given below.
S -> CC
C ->aC
C -> d
Construct a CLR parsing table for the above grammar
Iteration:

6




Find the Follow values:

$S' \rightarrow S\$$ Follow (S) = { $ }

$S \rightarrow C\,C$ (r1) Follow (C) = { $ ,a,d }

$C \rightarrow a\,C$ (r2)

$C \rightarrow d$ (r3)

Constructing the parsing table

| States | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | c | d | S | S | C |
| 0 | S3 | S4 | | 1 | 2 |
| 1 | | | Acc | | |
| 2 | S6 | S7 | | | 5 |
| 3 | S3 | S4 | | | 8 |
| 4 | r3 | r3 | | | |
| 5 | | | r1 | | |
| 6 | S6 | S7 | | | 9 |
| 7 | | | r3 | | |
| 8 | r2 | r2 | | | |
| 9 | | | r2 | | |

6

HOD/CSE



Signature of the Faculty