Name: Obaid-Ur-Rahman Siddiqui

Reg. No: L1S19BSCS0001

Section: (G1)

Compiler Construction (CC)

Assignment # 2

Submitted to: Sir Muhammad Usman Afzal

Due Date: 12-06-2022

privert the following complete grammar into LL(1) predictive

Function -> Type identifier (Arglist) | Compound Stmt

Arglist -> Arg / Arglist , Arg

Arg -> Type identifier

Declaration -> Type Identlist;

Type -> int I float

Identlist -> identifier, Identlist 1 identifier

Stmt -> WhileStmt | Rualue; | IfStmt | Compound Stmt | Declaration |;

While Stmt -> while (Rualie) Stmt

If Stmt -> if (Rualue) Stmt ElsePart

ElsePart \rightarrow else Stmt 1 ϵ

Compound Stmt → { Stmt List }

StmtList -> StmtList Stmt 1 E

Rualue -> Rualue Compare Mag 1 Mag

Compare → == 1 < 1 > 1 <= 1 >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >= 1 != 1 < >

Mag -> Mag + Term / Mag - Term / Term

Term -> Term * Factor | Term / Factor | Factor

Factor -> (Mag) / identifier / number

Willym do buy

Just I foot in 1911

of a without in the second

Industries of Type I deather

Answer:

There are 3 steps to convert the grammar into LL(1) predis

- Remove Ambiguity
- · Remove Left recursion
- · Left Factoring.

Step-01: Remaile Ambiguity.

There is no ambiguity in the given grammar so we mave to

Step-02: Remove Left Recursion:

Function -> Type identifier (Arglist) Compound Start

Ang Anglist

Arglist -> , Arg Arglist 1 E

Arg -> Type identifier

Declaration -> Type IdentList ;

Type -> int / float

Identlist -> identifier, Identlist / identifier

Stmt -> While Stmt | Rushue ; | Ifstmt | Compound Stmt | Declaration 1;

While Start -> while (Rualue) Start

If Stort -> if (Rualue) Stort ElsePart

Else Part \rightarrow else Stort / ϵ

Compound Start -> { Start List }

Just → E Stont List'

**List' → Stont Stont List' | E

**Julue → Mag Rualue' (3) Carronal Stant . . ! Clint List ? on is a colonial 3 1 'distribute the committee' 1 & which for Reduct Rualue -> Compare May Rualue 1 E Mag -> Term Mag' Mars & John May Mag' -> + Term Mag' | - Term Mag' | E - I transform ! - - - -Term -> Factor Term -> Minimal Grammar Term -> * Factor Term 1 / Factor Term 1 & Tenn -> * Each To Factor -> (Mag) / identifier / number Factor - S. Mary J. Ja. Step-03: Left Factoring Function -> Type identifier (Arglist) Compound Start Arglist -> Arg Arglist' 7 . no montalia Arglist -> , Arg Arglist 1 Ex San he reveded ittil predicting minimal Arg -> Type identifier - pullered . Declaration -> Type IdentList; I Mag - Tenm Mag Type -> int I float E. Mag - + + Term Mag Identilist -> identifier Identilist .. Mag - - Term Mag Identlist → , Identlist / E Stort -> While Stort | Rudue ; / If Stort | Compound Stort | Declaration 1; While Stont -> while (Rualue) Stont From - Factor Term If Stmt o if (Rualue) Stmt Else PartElse Part o else $Stmt I ext{ } E$ "Town - w Factor Town

long, long Compound Start $\rightarrow \{Start List\}$ Start List -> E Start List' Statlist -> Start Startlist 1 E Rualue -> Mag Rualue' in the second Rualue' -> Compare Mag Rualue' 1 E - 1 - 1 = = - - medium Compare -> == 1<1>1<=1>=1!=1<> Mag → Term Mag' King to & of Town 1. Mag' -> + Term Mag' | - Term Mag' | E -> Minimal Grammar Term -> Factor Term' Term' -> * Factor Term' 1 / Factor Term' / E Frather - May 1; Factor -> (Mag) / identifier / number Therefore, the quien grammar has been converted to LL(1) predictive Grammar. would the hope stylist A Implementation Part A. I distant prin . . . distant The corrected LL(1) predictive minimal grammar is given as follows: -Dickoralies - Type Identick : 1. Mag -> Term Mag' Type - wit I flood 2. Mag' -> + Term Mag' Identicate - identifier Intentials 3. Mag' → - Term Mag' 3 / Siddhabi . " - " duidab I E Wilhite Start I Rushue: 1 Tistant I 4. May -> E -> while (Rushuc) that
if (Rushuc) that
if (Rushuc) that
-> else is 5. Term -> Factor Term' 6. Term' -> A Factor Term'

1 > 1 Factor Term

 $: 1 \text{ orm}' \rightarrow \mathcal{E}$

9. Factor -> (Mag)

10. Factor → identifier

11. Factor → number

We first find the FIRST and FOLLOW sets of all

Variable

Mag

Mag

FIRST { C, identifier, number }

{+,-,e}

Term { C, identifier, number }

Term' [*,1, E]

Factor { C, identifier, number } Now we inorto FOLLOW

{\$, }}

{\$,5}

{+,-,\$,)}

{+,-,\$,}}

[+,1,+,-,\$,)}

Variable	identifier	' u		L(1)	pa	rsing	Tale	le as	follows
Mag	1	number	+		*	/	C)	R
			Skip	Skip	ship	skip	1	PoP	PoP
Mag'	Skip	Skip	2	3	Skip	Skip	Skip	4	4
Term	5	5	PoP	PoP	Skip	Skip	5	PoP	PoP
Term' Factor	Skip	Ship	8	8	6	7	skip	8	8
	10	11	PoP	PoP	PoP	PoP	9	PoP	PoP