Compiler Construction

Assignment-2

22L-6795 6F

M.Tayyab

working

=> Arg List -> Arg | ArgList, Arg }-> contains left recursion

(removing)

Arg list -> ~~Ag~~ Arg ArgList'

ArgLis List' -> , Arg ArgList' | ^

=> Identlist -> identifier, IdentList | identifier

(contain ambiguity perfoming left factoring)

Ident list -> identifier IdentList'

Ident list' -> , Identlist | ^

=> Stmt List -> StmtList Stmt | ^

(contain left recursion now removing)

Stmt list -> StmtList'

StmtList' -> Stmt StmtList' | ^

=> Rvalue -> Rvalue Compare Mag | Mag

(contain left recursion now removing)

Rvalue -> Mag Rvalue'

Rvalue' -> Compare Mag Rvalue' | ^

=> Mag → Mag + Term | Mag - Term | Term

1 (it contain Ambiguity & left recursion, First perform left fator)

Mag → Mag Mag' | Term

Mag' → + Term | - Term } now

(now perform removing of left recursion)

1 Mag → Term Mag''

2 Mag'' → Mag' Mag'' | ∧ } → Final

3 Mag' → + Term | - Term

=> Term → Term * Factor | Term / Factor | Factor

(it contain ambiguity & left recursion, First left factoring)

Term → Term Term' | Factor

Term' → * Factor | / Factor

(now removing left recursion)

1 Term → Factor Term''

2 Term'' → Term' Term'' } → Final

3 Term' → * Factor | / Factor

=> Type → Adadi | Ashriya | Harf | Math | Mantiqi

(it contain ampiguity perform left factoring)

Type → A Type' | Harf | Ma Type''

Type' → dadi | shriya

Type'' → th | ntiqi

=> Compare → == | < | > | <= | >= | != | <>

(it has Ambiguity now doing left factoring)

Compare → < Comp' | > Comp'' | == | !=

Comp' → = | > | ∧

Comp'' → = | ∧

Final Solution

Function → Type identifier (ArgList) CompStmt

ArgList → Arg ArgList'

ArgList' → , Arg ArgList' | ^

Arg → Type identifier

Declaration → Type IdentList::

Type → A Type' | Harf | Ma Type''

Type' → dadi | shriya

Type'' → tn | ntiqi

IdentList → identifier IdentList'

IdentList' → , IdentList | ^

Stmt → For Stmt | While Stmt | Expr:: |
If Stmt | CompStmt | Declaration | ::

For Stmt → for(Expr::OptExpr::OptExpr) Stmt

OptStmt → Expr | ^

While Stmt → while (Expr) Stmt

IfStmt → Agar(Expr) Stmt ElsePart

ElsePart → Wagarna Stmt | ^

CompStmt → {StmtList}

Stmt List → StmtList'

StmtList' → Stmt StmtList' | ^

Expr → identifier := Expr | Rvalue

Rvalue → Mag Rvalue'

Rvalue' → Compare Mag Rvalue' | ^

Compare → == | != | <Comp' | >Comp''

Comp' → = | > | ^

Comp'' → = | ^

Mag → Term Mag''

Mag'' → Mag' Mag'' | ^

Mag' → +Term | -Term

Term → Factor Term''

~~Term~~ Term'' → Term' Term''

Term' → * Factor | / Factor

Factor → (Expr) | identifier | number