

**EAST WEST UNIVERSITY**

Fall-2019

**Project Report**

Project Name: Making A grammar For Mini Compiler

Course code: CSE375

Course Title: Compiler Design

Section: 01

**Submitted To:**

Dr. Shamim H Ripon

Professor, Dept. Of CSE

**Submitted by:** Jannatul Naim

ID: 2017-2-60-029, Dept. of CSE

**Features Supported by the Grammar:**

* Header File
* Variable Declaration
* Function
* Conditional Statement

For and while loop

**Our grammar for mini compiler:**

grammar o ;

root : (headerf)+ mfunction (function)\* ;

headerf : '\*' 'start' '#' dtype '!p' '#' ;

dtype : 'stdio'

| 'stdlib'

| 'string' ;

function :vtype (LET)+ '(' (vtype LET)? ')' mainblock ;

mfunction : vtype 'main' '(' ')' mainblock ;

mainblock : '[' mainblockD ']' ;

mainblockD :( statement

| forstatement

| ifstatement

| whilestatement )+ ;

statement : (dec)+ (expr)\* ;

forstatement : foR (expr)\* ;

ifstatement : ifBlock ;

whilestatement : whilE (expr)\* ;

foR : 'for' '(' exp '.' exp '.' LET incdec ')' forblock ;

whilE : 'while' '(' exp ')' whileBlock ;

funC : (LET)+ '(' (term)? ')' '$' ;

forN : 'for' '(' exp '.' exp '.' LET incdec ')' nestedFor ;

forblock : '[' forblockD ']' ;

forblockD : (expr)\* forN (expr)\*

| (expr)\* ifBlock ;

whileBlock :'[' whileBlockD ']' ;

whileBlockD : (expr)\* ifBlock ;

nestedFor : '[' nestedForD ']' ;

nestedForD : (expr)\* ifBlock ;

ifBlock : 'if' '(' exp ')' ifElseBlock 'else' ifElseBlock ;

ifElseBlock : '[' ifElseBlockD ']' ;

ifElseBlockD: (expr)+ ;

exp : exp binary exp

| exp relational exp

| exp logical exp

| '(' exp ')'

| term operand term

| term ;

dec : vtype (LET (',')?)+ '$' ;

expr : exp '$’ ;

binary : '+' | '-' | '\*' | '/' ;

relational : '=' | '!=' | '>' | '>=' | '<' | '<=' | '==' ;

logical : '&' | '|' ;

incdec : '++'| '--' ;

operand : '+=' | '-=' ;

term : LET | DIG ;

DIG :[0-9]+ ;

LET : [a-zA-Z]+ ;

vtype : 'int' | 'bool' | 'char' | 'float' ;

WS : [ \t\r\n]+ -> skip ;

**Sample Input and Parsing Tree:**

**1st one:**

\*start#stdio!p#

\*start#stdlib!p#

int main()[

int a,b,c $

a=b+c $

for(a=0 . a<10 . a++)[

b=b+10 $

for( a=0 .a<10 . a++)[

c=c+1 $

if(b>c)[

d=D+2 $

]

else [

c=c-1 $

]



**2nd one:**

\*start#stdio!p#

\*start#stdlib!p#

int main()[

int a,b,c $

a=b+c $

while(a<10)

[

c=c+1 $

if(b>c)[

d=D+2 $

]

else [

c=c-1 $

]

]

]



**Sample Input And Parsing Tree With Error:**

\*start#stdio!p#

\*start#stdlib!p#

int main()[

while(a<10)

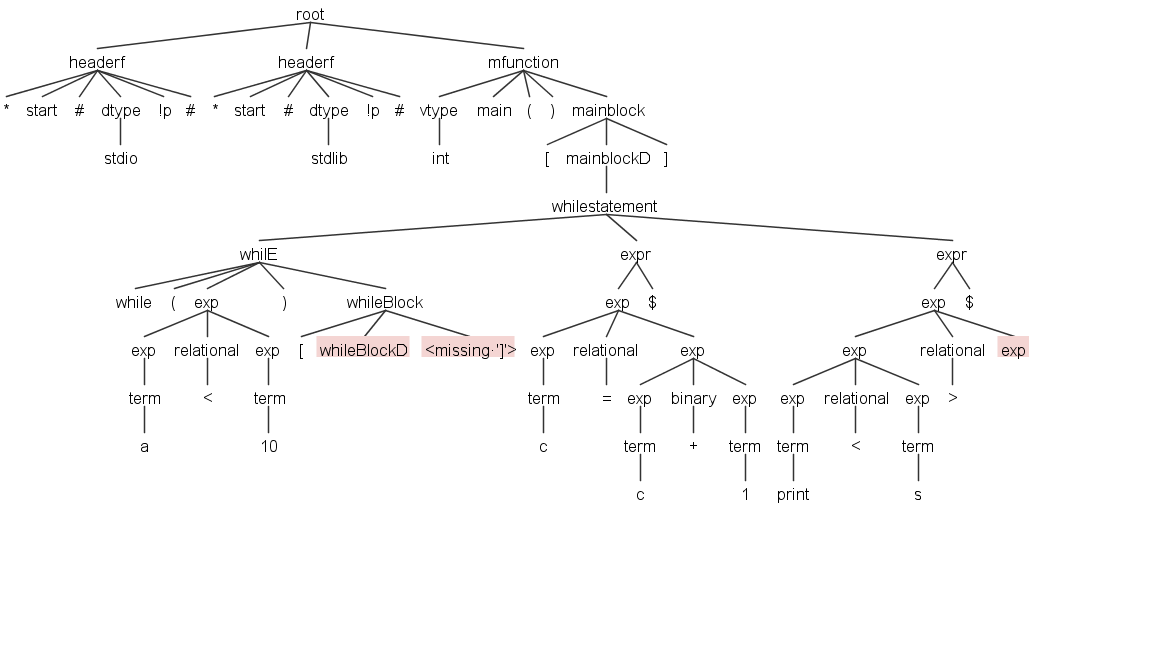
[

c=c+1 $

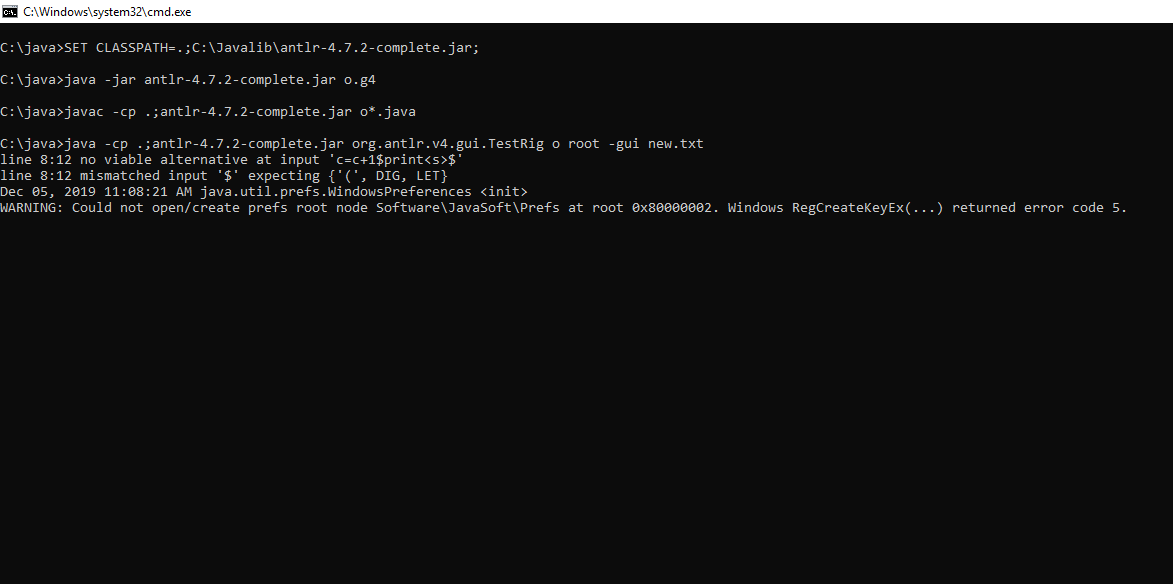
print < s > $

]

]



**Error in command promt:**



**Conclusion:**

After complete this project, we can learn how to making a grammar.