Ex-no: 5
08-10-2024

M. Rohith
3122 21 5001 085

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UCS 2702- Compiler Design Lab
Programming Assignment-5
- Implementation of Syntax checker using Lex and Yacc Tools

Develop a Syntax checker to recognize the following statements by writing suitable grammars

Assignment statement
Conditional statement
Looping statement
Declaration statement

INPUT

```
int i, a;
i = 0;
a=5;
while (i < 10)
{
if ( i<a)
i = i + 1;
else
i = i – 1;
}
```

OUTPUT

Syntactically correct

Program code:

parser.l

```
%{
#include "parser.tab.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
%}

/* Definitions for regular expressions */
identifier  [a-zA-Z_][a-zA-Z0-9_]*
number      [0-9]+
whitespace  [ \t\n]+

%%

"int"         { return INT; }
"if"          { return IF; }
"else"        { return ELSE; }
"while"       { return WHILE; }

{identifier}  {
   yylval.str = strdup(yytext);
   return IDENTIFIER;
}
{number}      {
   yylval.num = atoi(yytext);
   return NUMBER;
}

";"           { return SEMICOLON; }
","           { return COMMA; }
"="           { return ASSIGN; }
"=="          { return EQ; }
"!="          { return NEQ; }
"<"           { return LT; }
">"           { return GT; }
"<="          { return LE; }
">="          { return GE; }
"+"           { return PLUS; }
"-"           { return MINUS; }
"*"           { return MULTIPLY; }
"/"           { return DIVIDE; }

"("           { return LPAREN; }
")"           { return RPAREN; }
"{"           { return LBRACE; }
"}"           { return RBRACE; }

{whitespace}  { /* ignore whitespace */ }
```

```
.           {
    printf("Invalid character: %s\n", yytext);
}

%%

int yywrap() {
    return 1;
}
```

parser.y

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void yyerror(const char *s);
int yylex();

%}

%union {
    char *str;
    int num;
}

%token <str> IDENTIFIER
%token <num> NUMBER
%token INT IF ELSE WHILE
%token ASSIGN EQ NEQ LT GT LE GE PLUS MINUS MULTIPLY DIVIDE
%token SEMICOLON COMMA LPAREN RPAREN LBRACE RBRACE

%%

// Grammar rules
program:
    statements
    ;

statements:
    statement
    | statements statement
    ;

statement:
    declaration_statement
    | assignment_statement
    | conditional_statement
    | looping_statement
    ;
```

```
declaration_statement:
    INT identifier_list SEMICOLON
    ;

identifier_list:
    IDENTIFIER
    | identifier_list COMMA IDENTIFIER
    ;


assignment_statement:
    IDENTIFIER ASSIGN expression SEMICOLON
    ;

expression:
    expression PLUS term
    | expression MINUS term
    | term
    ;

term:
    term MULTIPLY factor
    | term DIVIDE factor
    | factor
    ;

factor:
    IDENTIFIER
    | NUMBER
    | LPAREN expression RPAREN
    ;

conditional_statement:
    IF LPAREN condition RPAREN statement
    | IF LPAREN condition RPAREN statement ELSE statement
    ;

condition:
    expression relational_operator expression
    ;

relational_operator:
    EQ | NEQ | LT | GT | LE | GE
    ;

looping_statement:
    WHILE LPAREN condition RPAREN LBRACE statements RBRACE
    ;

%%
```

```
// Error handling function
void yyerror(const char *s) {
    fprintf(stderr, "Error: %s\n", s);
}

int main(void) {
    if (yyparse() == 0) {
        printf("Syntactically correct\n");
    } else {
        printf("Syntax error detected\n");
    }
    return 0;
}
```

input.txt

```
int i, a;
i = 0;
a=5;
while (i < 10)
{
    if ( i<a)
    i = i + 1;
    else
    i = i - 1;
}
```

Output: