**Parser Design Document**

**Ahad Lodhi - 2020042**

**1. Introduction**

The parser is designed to handle a simple expression language that supports basic arithmetic operations, variables, and parentheses. The implementation consists of two main components: the Lex code for lexical analysis and the Yacc code for syntax analysis.

**2. Lexical Analysis (Lex)**

**2.1 Lexical Specification**

The Lex code specifies the rules for tokenizing the input source code. Tokens include numbers, variables, and operators.

[0-9]+ { yylval.num = atoi(yytext); return NUMBER; }

[a-zA-Z]+ { yylval.str = strdup(yytext); return VARIABLE; }

[ \t\n] ; // Skip whitespace

[-+\*()] { return yytext[0]; }

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

**3. Syntax Analysis (Yacc)**

**3.1 Grammar Specification**

The Yacc code defines the grammar rules for the expression language. It specifies how expressions, terms, and factors are structured.

expr: term exprbar

;

exprbar: '+' term exprbar { printf("+ "); }

| '-' term exprbar { printf("- "); }

| /\* empty \*/ { printf("\n"); }

;

term: factor termbar

;

termbar: '\*' factor termbar { printf("\* "); }

| /\* empty \*/ { }

;

factor: NUMBER { printf("Number= %d ", $1); }

| VARIABLE { printf("Variable = %s ", $1); free($1); }

| '(' expr ')' { }

;

**3.2 Abstract Syntax Tree (AST)**

Although not explicitly defined in the provided code, the actions in the Yacc rules can be extended to build an Abstract Syntax Tree (AST). This AST can be used for further processing or code generation.

**4. Integration**

The main function integrates the Lex and Yacc components, initiating the parsing process.

int main() {

yyparse(); // Start the parsing process

return 0;

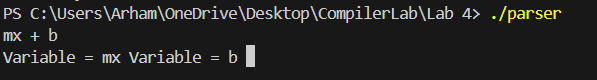
}

**5. Conclusion**

The parser successfully tokenizes and parses expressions, generating output that reflects the structure of the input. Further enhancements may include extending the grammar to support additional features, error handling improvements, or integrating the generated AST for more advanced applications.

Remember to update this document as the parser evolves and additional features are implemented.

**Test Cases:**

****

**A black screen with white text

Description automatically generated**