King Saud University

College of Computer and Information Sciences

Computer Science Department

CSC 340

Programming Language and Compilation

Student name	ID
Aliyah Aljarallah	443201214
Shoug Alsaleem	443200641
Shahad aloushan	443201080
Arwa Alfarhood	441200604

Project Description

This project is a parser for the Tiny language, built using Flex and Bison.
The parser checks whether a Tiny program conforms to the language's grammar rules and provides helpful syntax error messages when violations occur.

Prerequisites

To compile and run this project, make sure the following tools are installed:

- Flex – Lexical analyzer generator

- Bison – Parser generator (like Yacc)

- GCC – C language compiler

How to Compile and Run

Generate Parser Files Using Bison

bison -v -d --file-prefix=y tiny_l.y

This command produces:

- y.tab.c: The parser implementation

- y.tab.h: Header file with token definitions

- y.output: Debug file showing the grammar and any conflicts

Working Examples

Example 1:

```
program test;
x: integer;
beginprogram
x := 5 + 3;
endprogram
```

Execution Command:

./parser < example.tiny

Output:

(base) alyaaljarallah@Alyas-MacBook-Pro Desktop % ./parser < example.txt

Variable declaration: x: integer
Assignment operation: x := expression
Production used: program -> program_declaration variable_declarations statement_list endprogram

Example 2:

```
program mathprog;

a: integer;

b: integer;

beginprogram

a := 7 * 2;

b := a + 10;

endprogram
```

Output:

(base) alyaaljarallah@Alyas-MacBook-Pro Desktop % ./parser < example.txt

Variable declaration: a: integer

Variable declaration: b: integer

Assignment operation: a := expression

Assignment operation: b := expression

Production used: program -> program_declaration variable_declarations statement_list endprogram

Example 3:

```
program arraytest;

arr: array(10) of integer;
```

beginprogram

arr := 4;

endprogram

Output:

(base) alyaaljarallah@Alyas-MacBook-Pro Desktop % ./parser < example.txt

Variable declaration: arr: integer

Assignment operation: arr := expression

Production used: program -> program_declaration variable_declarations statement_list endprogram

Examples with Syntax Errors:

Example 1:

program x;

a integer;

beginprogram

a := 1 + 2 * 3;

endprogram

Execution Command:

./parser <error_example.tiny

Output:

(base) alyaaljarallah@Alyas-MacBook-Pro Desktop % ./parser < example.txt

Syntax error at line 2: invalid declaration (missing ':')

Assignment operation: a := expression

Production used: program -> program_declaration variable_declarations statement_list endprogram

Example 2:

```
program exprtest;
x: integer;
beginprogram
x := 5 + ;
endprogram
```

Output:

```
(base) alyaaljarallah@Alyas-MacBook-Pro Desktop % ./parser < example.txt
Variable declaration: x: integer
Syntax error at line 4: incomplete expression
Production used: program -> program_declaration variable_declarations statement_list endprogram
```

Example 3:

```
program test;
x: integer
beginprogram
x := 5 + ;
endprogram
```

Output:

```
(base) alyaaljarallah@Alyas-MacBook-Pro Desktop % ./parser < example.txt
Syntax error at line 2: ';' expected
Syntax error at line 4: invalid declaration
```