**15 CSE385 – COMPILER DESIGN LAB – LAB SHEET 4**

For this lab, you will use a parser generator called “Constructor of Useful Parsers” (CUP for short). CUP is a system for generating LALR parsers from simple specifications. To get started:

***Download cup-11a.jar from your AUMS account, extract the jar file, and place the folder java\_cup in your working folder.***

**EXAMPLE 1:** A grammar for simple declaration statement. Carefully go through the rules and make sure you understand the significance of every line in the grammar. Follow the step-by-step instructions below to create the grammar, compile and execute:

1. Save the below file as **declaration.**

**JLEX FILE**

import java\_cup.runtime.Symbol;

import java\_cup.runtime.Scanner;

%%

%cup

%eofval{

System.exit(0);

%eofval}

%%

";" {System.out.println("LA "+yytext());return new Symbol(sym.SEMI);}

"," {System.out.println("LA "+yytext());return new Symbol(sym.COMMA);}

" " {System.out.println("LA "+yytext());return new Symbol(sym.SPACE);}

[0-9]+ {System.out.println("LA "+yytext());return new Symbol(sym.INT,new Integer(yytext()));}

"int" {System.out.println("LA "+yytext());return new Symbol(sym.INT);}

"string" {System.out.println("LA "+yytext());return new Symbol(sym.STRING);}

[a-z][a-z0-9]\* {System.out.println("LA "+yytext());

return new Symbol(sym.ID,new String(yytext()));}

[\n\r] {System.out.println("LA"+ "EOF");return new Symbol(sym.EOFILE);}

1. Type the following grammar and save the file as **declaration.cup**  for recognizing simple declaration statements in a text editor.

**CUP FILE**

import java\_cup.runtime.\*;

scan with {: return getScanner().next\_token(); :};

terminal INT,STRING,SEMI,COMMA,ID, SPACE, EOFILE;

non terminal prog, stmt, decln, varlist, type, s;

s::=prog {: System.out.println("Valid declaration"); :}

EOFILE{:System.exit(0);:} ;

prog::=prog stmt |stmt ;

stmt::=decln;

decln::=type SPACE varlist SEMI;

type::=INT | STRING ;

varlist::=ID COMMA varlist | ID ;

1. At the command prompt, type the following command to generate the Jlex & Parser code

in Java as

CMD> jlex declaration

CMD>cup declaration.cup

If the above step completed successfully, you will find three files in your working directory: Parser.java, Sym.java, declaration.java

1. Type the following and save the file as **Main.java.**

import java.io.\*;

public class Main

{ public static void main(String[] args)throws Exception

{

parser p = new parser(new Yylex(new FileReader("input"))); p.parse(); } }

1. Create the input file with the file name **input** which contains a valid declaration statement.

Eg: int a,b,c;

1. Compile the \*.java files. The corresponding .class files will be created.

CMD> javac \*.java

1. Run the Main Program as

CMD>java Main

If the input is correct as per the grammar definition - you will get “Valid declaration” as output. It merely means the input is accepted. For incorrect input, you will get errors. Try typing incorrect syntax or some junk text to get error messages.

**Exercises:**

1. Extend the same grammar to accept the declarations of the form:
2. int a,b,c;
3. float a,b,sum;
4. char ch,ch1;
5. Given below are some exercises that enhances the simple declaration statement to recognize a wide variety of other declarations.
6. Multiline declaration: Extend the syntax rules to accept declaration statements with newline in between.

Example: int a,b,c;

int c,d;

1. Optional Definition: Extend the syntax rules to accept declaration statements of the form: var int tokens = 2, tip, args = 53; var string xyz = “hello”;
2. Floating point: Extend the lexical and syntax rules to recognize floating point numbers (2.345) as FLOAT and in declaration statement.

Example: var float time = 12.34;

1. Constant Declaration: Extend the syntax rules to recognize declaration of constants of the form: const int pi = 3.14;. Note, constant declaration can be done one at a time and definition during declaration is compulsory. i.e. const string xyz; is invalid whereas const string xyz = “hello”; is valid.
2. NULL value: Extend your syntax rules to accept nil as a value during declaration

Example: var int i = NULL; var string s = NULL;

f. Array Declaration: Extend your rules to recognize array declarations of the form: int value[10]; string colours[5];

1. Array Initialization: Extend your rules to recognize array initializations of the form: int value[10] = {1, 25, 43, ...... }; string colors[5] = {“red”, “green”, “blue”, “orange”, “yellow”};
2. Add rules to recognize assignment statements.

a. Assignment statements of the form id = expr;. Note that the precedence should be taken care of. \* and / has higher precedence than + and - while defining expr.

b. Modify the assignment rule to allow assignments of the form id.id = expr; or id.id.id.id = expr;.

Example: student.age = 15;

c. Modify further to allow assignments of the form id[num] = expr;

1. Extend rules to recognize if construct. Note that, else-part is optional.
2. Rules of the form: if (condition) statements else statements

condition: ID comparator expr

comparator: < | > | <= | >= | == |! =

1. Function Declaration

F 🡪 type FName LPAREN Arg RPAREN SEMI

FName - > ID

Arg 🡪 Arg, ID

Arg 🡪 ID