Comp-2140: Lab Assignment 3: Parser Generation

1 Submission

You should submit and test your programs at the submission site (you can click the left highlighted words to go to the submission site). The due date is Friday. It is worth 8 points. We will use your last submission for your final marks for this assignment.

You must test your program on our sample inputs and other cases devised by you before submission. Passing the sample inputs will not guarantee a high mark. You must fully understand the grammar and implement that grammar correctly so that your parser can cope with all situations.

2 Assignment Specification

This assignment is to parse Tiny programs using JavaCUP. Here are some useful links for the assignment, including the EBNF for Tiny language. You need to write a JavaCUP file and a JLex file so that a parser for Tiny language can be generated. The parser will be able to tell whether a Tiny program is syntactically correct. You need to rewrite EBNF grammar into CFG that is acceptable by JavaCUP. Comments in Tiny program should be thrown away in the scanner.

We will run the following commands to generate the scanner and the parser. You can modify the lex file you created in assignment 2.

```
>java JLex.Main A3.lex
>java java_cup.Main -parser A3Parser -symbols A3Symbol < A3.cup
>javac A3.lex.java A3Parser.java A3Symbol.java A3User.java
>java A3User
```

The program A3User invokes the parser. It is defined as below:

```
import java.io.*;
class A3User {
  public static void main(String[] args) throws Exception {
       File inputFile = new File ("A3.tiny");
       A3Parser parser= new A3Parser(new A3Scanner(new FileInputStream(inputFile)));
       Integer result =(Integer)parser.parse().value;
       FileWriter fw=new FileWriter(new File("A3.output"));
       fw.write("Number of methods: "+ result.intValue());
       fw.close();
  }
}
```

If your lex and cup files are correct, all of those command and especially A3User will run smoothly without any error report, and an A3.output file will be created which should consists of one line as follows:

```
Number of methods: numberOfMehtodsInA2Input
```

If your programs are not correct, during the process there will be some error messages. Here is a sample A3.tiny file and A3.output file. For incorrect Tiny programs such as A31.tiny and A32.tiny, your parser will report an error and no `A3.output` file is generated.

Note that you don't need to write any Java programs. The parser and the scanner are generated from your cup and lex specifications. Also, we will test your program on our own data.

3 What to submit

You need to turn in two files: the JLex file, named `A3.lex`, which can be used to generate the scanner; the javaCUP file, named `A3.cup`, which can be used to generate the parser.

4 Installing JavaCup

Download the JavaCup tar file by right-clicking on the link here. Save that file in your 2140 directory. It is a .tar file. You can run the following to extract the files to your 2140 directory:

```
tar -xvf  javaCup.tar
```

Here -x means extraction, -v means verbose, and -f means from a file. It will create the java_cup directory, and extract the classes into that directory. The result of running the command will look like this:

```
2140$ tar -xvf javaCup.tar
x java_cup/CUP$parser$actions.class
x java_cup/Main.class
x java_cup/Main.java
x java_cup/action_part.class
x java_cup/action_part.java
x java_cup/action_production.class
x java_cup/action_production.java
x java_cup/assoc.class
x java_cup/assoc.java
x java_cup/emit.class
x java_cup/emit.java
...
```

Unlike JLex, java_cup has java files as well as class files, hence you do not need to compile them. To run your javaCup, you go to the directory that is one layer above java_cup (for instance 2140), type

```
java java_cup.Main < yourCupFile
```

The above command will generate the parser.

Use scripts

Now you need to run a sequence of commands often. Instead of typing the long commands every time, you can save the commands in a file (script) and run the sequence of commands by running the script.

A sample script for calc example is as follows:

```
java JLex.Main calc.lex
java java_cup.Main -parser CalcParser -symbols CalcSymbol calc.cup
javac calc.lex.java
javac CalcParser.java CalcSymbol.java CalcParserUser.java
java CalcParserUser
```

You can create a file named `calc.bat` to contain those lines, or you can save the bat file by right-clicking the file name calc.

To run the script, you can type "calc" in Windows.

In Unix/Linux/Ubunto/OS X, you should type "./calc.bat", the entire file name, to run the script. On the other hand, In unix you do not need to have the `.bat` file extension. Any file name works. In addition, you need to change the access mode by typing

```
chmod 755 calc.bat
```

That means to change the access mode to executable and readable. Otherwise it can not be executed.

5 Steps to work on the assignment

1. Run calc.lex and calc.cup example.
2. Tansform calc.lex and calc.cup to A3.lex and A3.cup. You need to change every occurrence of Calc to A3 in the lex, cup, and script file. Make sure that you can run A3. In this step, try to understand how A3.lex and A3.cup work together. For example, in the following calc.lex file, CalcScanner needs to be changed to A3Scanner, CalcSymbol needs to be changed to A3Symbol.

```
import java_cup.runtime.*;


%implements java_cup.runtime.Scanner
%type Symbol
%function next_token
%class CalcScanner
%eofval{ return null;
```

```
%eofval}

IDENTIFIER = [a-zA-Z_][a-zA-Z0-9_]*
NUMBER = [0-9]+

"+" { return new Symbol(CalcSymbol.PLUS); }
"-" { return new Symbol(CalcSymbol.MINUS); }
"*" { return new Symbol(CalcSymbol.TIMES); }
"/" { return new Symbol(CalcSymbol.DIVIDE); }
{NUMBER} { return new Symbol(CalcSymbol.NUMBER, new Integer(yytext()));}
\r|\n {}
. {}
```

3. Remove the evaluation function. This assignment won't evaluate the values of expressions, hence we should remove that functionality. To do so, we need to remove the RESULT assignment statements in the cup file:

```
terminal          PLUS, MINUS, TIMES, DIVIDE, LPAREN, RPAREN;
terminal Integer  NUMBER;
non terminal Integer expr;
precedence left PLUS, MINUS;
precedence left TIMES, DIVIDE;
expr      ::= expr:e1 PLUS expr:e2 {: RESULT = new Integer(e1.intValue()+ e2.intValue()); :}
            | expr:e1 MINUS expr:e2  {: RESULT = new Integer(e1.intValue()- e2.intValue()); :}
            | expr:e1 TIMES expr:e2  {: RESULT = new Integer(e1.intValue()* e2.intValue()); :}
            | expr:e1 DIVIDE expr:e2  {: RESULT = new Integer(e1.intValue()/ e2.intValue()); :}
            | LPAREN expr:e RPAREN {: RESULT = e; :}
            | NUMBER:e {: RESULT= e; :}
            ;
```

That is, your A3.cup file should look like this:

```
terminal          PLUS, MINUS, TIMES, DIVIDE, LPAREN, RPAREN;
terminal Integer  NUMBER;
non terminal Integer expr;
precedence left PLUS, MINUS;
precedence left TIMES, DIVIDE;
expr      ::= expr PLUS expr {:  :}
            | expr MINUS expr  {: :}
            ;
```

Note that labels like e1 and e2 are removed, along with the action code inside the brackets "{:" and ":}" .

4. Expand A3.cup file. Do not add many rules at once. Each time add one more rule, correct possible JavaCup errors or JLex errors, and test your newly added feature accordingly. For example, you can add an ID as a valid expression, then you should create a test cases that has and variable in expression, and test your code on the new test case.

6 Marking scheme

```
yourMark=0;
if ( A3.lex and A3.cup  files not submitted or file names are incorrect) return;
if (  A3.lex.java && A3Parser.java && A3Symbol are generated after the second command) {
   if ( generated programs are compiled correctly &&
        A3Scanner.class, A3Parser.class and A3Symbol.class are generated) {
            for (each of the 12 test A3.tiny) {
                 if (A3.tiny is valid && A3.output is generated && method count is correct)  {
                                yourMark+=0.666;
                 if (A3.tiny is not a correct Tiny program && A3.output is not generated )
                                yourMark+=0.666;
             }
         }
     }
 }
yourMark+= (size of your lex and cup file<350)?0.5*350/fileSize:0;
for (each day of your late submission)  yourMark=yourMark*0.7;
```