Keenan Thompson

7 February 2017

CS 536 - Professor Thomas Reps

Homework 2

Question 1

1) Incorrect
   - Does not allow for comments to contain asterisks, slashes and left or right parens.

     Example: `/* this is (good) / but not accepted */`
     Example: `/* this *is* /good/ but not accepted */`

2) Incorrect
   - Does allow `*/` within a comment. It's somewhat ambiguous how it would be processed, nonetheless it technically accepts the following as a single comment, when it should not:

     Example: `/* bad comment */ */`
     Example: `/* */ /* */`
   - Does not allow for multiple line comments. This is because `(.)*` doesn't match the new-line character.

     Example:
     ```
     /*
      * comment
      */
     ```

3) Incorrect
   - Does not allow non-empty comments to have a single asterisk at the end because `{STAR}+` is present twice in the pattern.

     Example: `/* This is a good comment. */`

Example: /* *This is a good comment.* */

4) Incorrect
   - Does not allow empty comments because **([^*]|[^/])+** indicates at least one character is needed in the body.

     Example: /**/
   - Does allow for ***/** within the comment body. This is because */ satisfies **([^*]|[^/])+** by first selecting a *****, then **/**.

     Example: /* */ */

5) Incorrect
   - Does not allow for asterisks before the end of the comment. Although any number of stars may precede the final ***/**, you can't have an asterisk and a non asterisk following it in the body.

     Example: /* this is not *accepted* */
     Example: /***** this isn't either */
   - Does allow for comments to have trailing slashes. Those slashes should not be perceived as part of the comment.

     Example: /* this is accepted *////

6) Correct, this matches all well-formed comments.

Question 2

Note: This prints the line number of each call to myFunc. The line number is one-based, unlike the zero-based value given by yyline. It prints the number of calls at the end.

I approached this by matching a single character that isn't a valid Java identifier (i.e. whitespace, semicolon, left brace) followed by "myFunc", followed by optional whitespace, followed by a left parenthesis.

```
// User code not needed.
%%

NON_JAVA_IDENT = [^a-zA-Z0-9_$]
WHITESPACE = [ \t\n]
MY_FUNC = ("myFunc"{WHITESPACE}*"(")
MY_FUNC_PREFIX = ({NON_JAVA_IDENT}|{WHITESPACE})
MY_FUNC_REF = {MY_FUNC_PREFIX}{MY_FUNC}

%implements java_cup.runtime.Scanner
%function next_token
%type java_cup.runtime.Symbol

%eofval{
  System.out.println("Total myFunc calls: " + myFuncReferences);
%eofval}
%{
  private int myFuncReferences = 0;
%}
%line
%%

{MY_FUNC_REF} {
```

```
    // Must correct for new line matching...
    int line = yytext().startsWith("\n")  ? (yyline + 2) : (yyline + 1);
    System.out.println("myFunc called at line " + line);
    myFuncReferences++;
}
.|{WHITESPACE} { }
```