COL 703: Logic for Computer Science

Assignment 1

Shruti Kumari
2018CS50420

Instructions for the execution of assignment:
To run the program, run the following commands in order:

To generate ast from flasl:

- In terminal, run 'ml-lex flasllex.lex' (to compile lex file)
- In terminal, run 'ml-yacc flasllex.yacc' (to compile yacc file)
- Open sml in interpreter mode by running 'sml'
- Run 'use "load-flasllex-ast.sml";' (combine the lex yacc with the sml files and generated sig file)
- Run 'val parsed = parseFile "arg-inp.sml";' (this will convert the flasl input from arg.sml and store it in the Argument type variable called parsed)

To save the ast generated to "arg.sml"

- Run 'use "ast2file.sml";' (to compile the file required to save parsed to arg-inp.sml)
- Run 'ast2file parsed;' (this will save parsed to arg.sml with the name of Argument being arg)

To generate flasl from ast:

- Run 'use "ast2flasl.sml";' (to compile the file required to do the task)
- Run 'use "arg.sml";' (to use the arg ast from arg.sml file)
- Run 'ast2flasl arg;' (this will convert arg back to flasl and store it to arg-out.flasl)

Explanation:
Lexing is done in the file flasllex.lex. Corresponding tokens are made and the text to be accepted is anything but [^\"\.\(\)\ \127\000-\031]. The line number is maintained using the variable linenum which is incremented on each encounter with ‘\n’. Similarly, a charsAbove variable is maintained which is reset to yypos on each encounter with ‘\n’. This keeps track of how many characters have appeared till the line just above so that the position of a character in its own line can be calculated using yyypos - charsAbove.

Parsing is done in the file flasllex.yacc using ML-Yacc. Since the content has to be an argument, Start points to either Conclusion or Proplist Conclusion, where Conclusion is THEREFORE Prop. Similarly, Proplist points to list of props, and prop is expression FULLSTOP. Similarly, we keep going down till we reach a string that is inside quotes. Now, to remove the extra spaces, we have taken the final string to be a list of strings inside quote, and once whole string list is gotten, it is concatenated using one single space.

Now, final parsed argument is stored to arg.sml using the functions defined in file ast2file.sml. For the second part, ast2flasl.sml is used to receive ast type arg from arg.sml, and now this arg is case matched to construct a string which is in flasl form. Now this final constructed string is stored in arg-out.sml. This concludes the assignment.