Name: Chitwan Goel
Roll No. 210295

CS335 Assignment 1 Report

1 Problem 1

1.1 File specifications

Inside the directory *problem1*, the file *prob1.l* contains the regular expressions and rules for the first problem.

1.2 Compilation Instructions

All commands to be executed inside *problem1* directory

```
$ flex prob1.l
$ g++ lex.yy.c -o lexer -ll
$ ./lexer < $[path_to_input_file]
```

Alternatively, the bash script *run.sh* can be executed in the following mannner to print the output on the screen.

```
$ chmod +x run.sh
$ ./run.sh $[path_to_input_file]
```

In the second step, there may be a change in the flag -ll depending on the system. On my WSL, it works with -ll flag and thus the script *run.sh* contains -ll flag in the command.

1.3 Output format

1. The first row of the program will be TOKEN COUNT LEXEME
2. Subsequent rows will contain information of each lexeme in the corresponding column.
3. All the rows except the header row is sorted lexicographically according to the LEXEME column.
4. The program will stop after reporting the first error message.

1.4 Error handling

1. Any invalid character or token is printed as "Invalid token/character at line number {{Line number}}". Since unicode characters create issue in printing, I have not printed the exact invalid character.
2. Any sort of invalid string is printed as "Invalid string {{Errorsome string}} at line number {{Line number}}" where Line number is the line at which the invalid string starts.
3. Following are the 4 cases for which invalid string error (Point 2.) is reported -
   (a) One or more single quote between two double quotes
   (b) One or more double quotes between two single quotes
   (c) Double quotes without their closing counterpart
   (d) Single quote without its closing counterpart
4. The curly braces in the 1st and 2nd point are meant to be representing placeholders and not to be printed by the code.
5. The program will terminate after catching the first error. It will print all the lexemes in the designated format and then the error statement before exiting.

1.5 Few Corner Cases

1. { marks the start of a comment until the first } is found. Nested comments, if any, are handled in the following manner:
   Suppose {{{}}} is in the input. So, {{{} is the comment and the other two } are counted as delimiters.

2. Case insensitive lexemes like keywords and operators are handled in the following manner:
   Suppose an input program has x occurrences of the keyword UNTIL in y uniquely different cases. Then, there will be y rows corresponding to each unique case but the count will be common for all, i.e., x.

3. Although hexadecimals are also case-insensitive, the count for them is not common as in the case for keywords or operators. For example, if there are *0xA* and *0xa*, then their corresponding rows will have count equal to 1.

2 Problem 2

2.1 File specifications

Inside the directory *problem2*, the file *prob2.l* contains the regular expressions and rules for the first problem.

2.2 Compilation Instructions

All commands to be executed inside *problem2* directory

```
$ flex prob2.l
$ g++ lex.yy.c -o lexer -ll
$ ./lexer < $[path_to_input_file]
```

Alternatively, the bash script *run.sh* can be executed in the following mannner to print the output on the screen.

```
$ chmod +x run.sh
$ ./run.sh $[path_to_input_file]
```

In the second step, there may be a change in the flag -ll depending on the system. On my WSL, it works with -ll flag and thus the script *run.sh* contains -ll flag in the command.

2.3 Output format

1. The first row of the program will be TOKEN COUNT LEXEME

2. Subsequent rows will contain information of each lexeme in the corresponding column.

3. All the rows except the header row is sorted lexicographically according to the LEXEME column.

4. The program will stop after reporting the first error message.

2.4 Error handling

1. Any invalid character or token is printed as "Invalid token/character at line number {{Line number}}".Since unicode characters create issue in printing, I have not printed the exact invalid character.

2. Any sort of invalid string is printed as "Invalid string {{Errorsome string}} at line number {{Line number}}" where Line number is the line at which the invalid string starts.

3. Following are the 2 cases for which invalid string error (Point 2.) is reported -
   (a) One or more single quote between two double quotes
   (b) Double quotes without their closing counterpart
4. The curly braces in the 1st and 2nd point are meant to be representing placeholders and not to be printed by the code.
5. The program will terminate after catching the first error. It will print all the lexemes in the designated format and then the error statement before exiting.

2.5 Few Corner Cases

1. ! marks the start of a comment only if it is the first non-blank character of a line. In this case, everything till the end of that line is ignored as a comment. Elsewhere, ! is counted as a special character.
2. Everything except string literal is case insensitive. Case insensitive lexemes are handled in the following manner:
   Suppose an input program has x occurrences of the keyword ELSE in y uniquely different cases. Then, there will be y rows corresponding to each unique case but the count will be common for all, i.e., x.
3. There is no restriction on leading zeroes in INT LITERALS or REAL LITERALS.