Makefile

In Makefile, we added new commands in test and cleantest.

test:

To test exceptions (illegal strings/tokens), we write the file exceptions.in which contains illegal strings. The file exceptions.out contains all the errors being printed out after tokens in exceptions.in coming to our scanner. Then we use diff command to test whether the exceptions.out is the same as exceptions.test to identify the correctness of our scanner.

To test eof, we use diff command to test whether the eof.out is the same as eof.test.

cleantest:

remove all out files (allTokens.out, exceptions.out, eof.out)

minim.jlex

In minim.jlex, we assigned some references refer to certain group of chars for easier implementing regular expression. Like the DIGIT already assigned by lecturer, we assigned LETTER, STRING, ESCAPE, BAD, NOTNL.
LETTER contains all chars from a to z & A to Z.
STRING contains all chars except (newline) \n, (double quote) \“, (backslash)\\. This is for writing regular expression for legal Strings.
ESCAPE defines all possible escape chars.
BAD defines all chars that would be bad if it has a backslash (\) just in front of it.
NOTNL contains all chars that do not represent a newline. This is for String’s regular expression. Since each line contains a sequence of chars that should be converted to corresponding tokens (we do not want combination of multiple lines), we do not want String’s regular expression to include a line ends with \n, and having double quote in next line.

We also implemented regular expressions that catch legal Strings and illegal Strings. For illegal Strings, we divide them into three main types (unterminated string literal, string with bad escaped character, and unterminated string literal with bad escaped character).

P2.java

Apart from the testAllTokens method provided by lecturer, we implement three extra tests. testExcepts is for testing all possible exceptions (errors), testChars is for testing whether CharNum.num is tracing correctly, and testEOF is for testing whether (eof) works correctly.