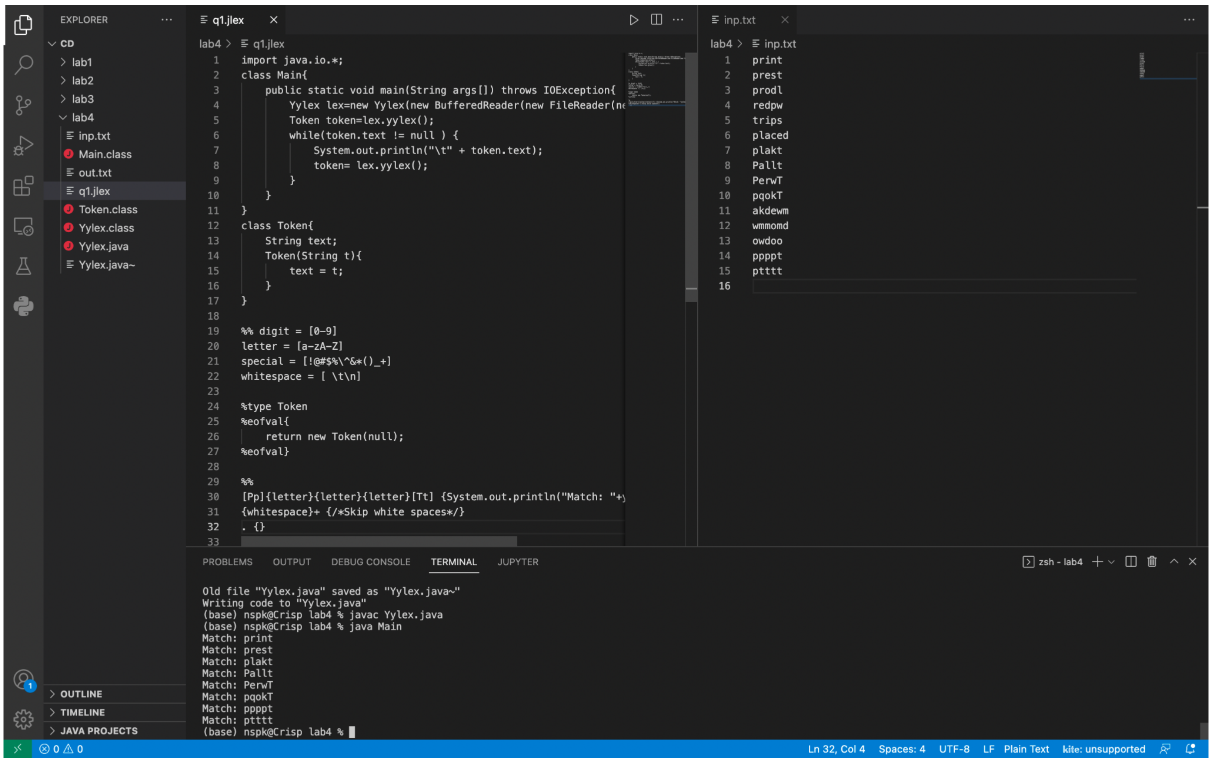
19CSE401 - Compiler Design

JLex Assignment

Lab sheet-4

Done By:
N Sai Pavan Krishna
AM.EN.U4CSE19347

A. Try the following JLex Program to recognize a 5 letter word which starts with P/p and ends with T/t.



q1.jlex

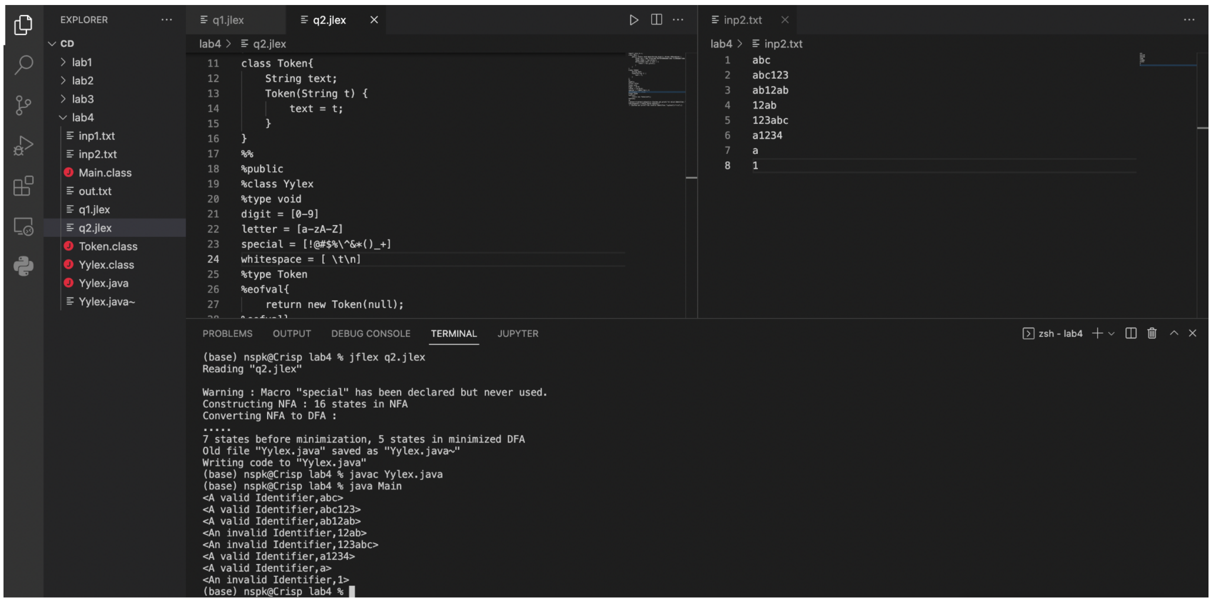
```
import java.io.*;
class Main{
    public static void main(String args[]) throws IOException{
        Yylex lex=new Yylex(new BufferedReader(new FileReader(ne
        Token token=lex.yylex();
        while(token.text != null ) {
            System.out.println("\t" + token.text);
            token= lex.yylex();
        }
    }
}
class Token{
    String text;
    Token(String t){
        text = t;
    }
}

%% digit = [0-9]
letter = [a-zA-Z]
special = [!@#$%\^&*()_+]
whitespace = [ \t\n]

%type Token
%eofval{
    return new Token(null);
%eofval}

%%
[Pp]{letter}{letter}{letter}[Tt] {System.out.println("Match: "+y
{whitespace}+ {/*Skip white spaces*/}
. {}
```

inp.txt

```
print
prest
prodl
redpw
trips
placed
plakt
Pallt
PerwT
pqokT
akdewm
wmmomd
owdoo
ppppt
ptttt
```

```
Old file "Yylex.java" saved as "Yylex.java~"
Writing code to "Yylex.java"
(base) nspk@Crisp lab4 % javac Yylex.java
(base) nspk@Crisp lab4 % java Main
Match: print
Match: prest
Match: plakt
Match: Pallt
Match: PerwT
Match: pqokT
Match: ppppt
Match: ptttt
(base) nspk@Crisp lab4 %
```

B. Try the following JLex Program to recognize an identifier which starts with a letter.



q2.jlex

```
class Token{
    String text;
    Token(String t) {
        text = t;
    }
}
%%
%public
%class Yylex
%type void
digit = [0-9]
letter = [a-zA-Z]
special = [!@#$%\^&*()_+]
whitespace = [ \t\n]
%type Token
%eofval{
    return new Token(null);
```

inp2.txt

```
abc
abc123
ab12ab
12ab
123abc
a1234
a
1
```

```
(base) nspk@Crisp lab4 % jflex q2.jlex
Reading "q2.jlex"

Warning : Macro "special" has been declared but never used.
Constructing NFA : 16 states in NFA
Converting NFA to DFA :
.....
7 states before minimization, 5 states in minimized DFA
Old file "Yylex.java" saved as "Yylex.java~"
Writing code to "Yylex.java"
(base) nspk@Crisp lab4 % javac Yylex.java
(base) nspk@Crisp lab4 % java Main
<A valid Identifier,abc>
<A valid Identifier,abc123>
<A valid Identifier,ab12ab>
<An invalid Identifier,12ab>
<An invalid Identifier,123abc>
<A valid Identifier,a1234>
<A valid Identifier,a>
<An invalid Identifier,1>
(base) nspk@Crisp lab4 %
```

1. Write JLex code for the following and output the token of the form **<token_name, lexem>**

    i. To recognize any Java identifier (a sequence of one or more letters and/or digits and/or underscores, starting with a letter or underscore. Token Name is **ID**

    ii. To recognize any Java identifier that does not end with an underscore. Token Name is ID

    iii. To recognize the keyword "if" in addition to identifiers. (Place the rule of "if" above the rule of identifier.) Token Name is **IF**

    iv. Move the "if" rule below that of identifier rule and check the effect on your input. Do you see any difference in the output?

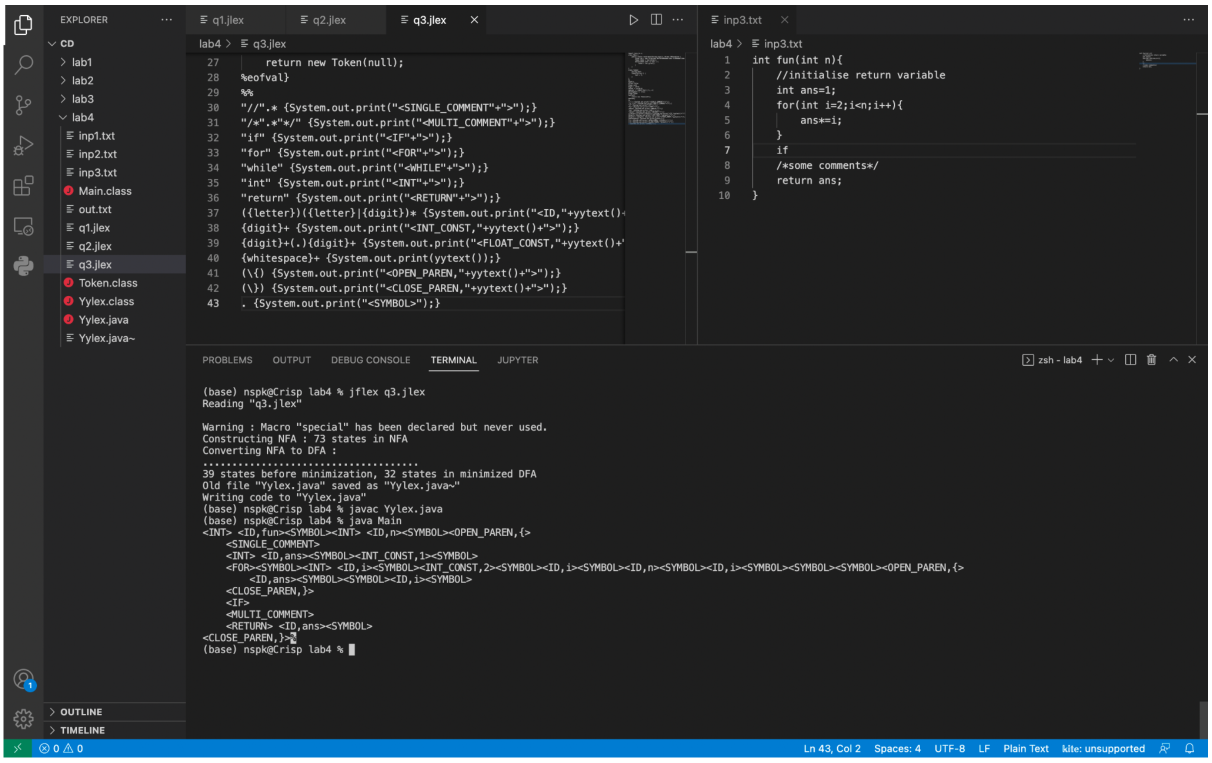
    v. Add the rule for other keywords, **for**, **while**, **do** and all types of parentheses in a similar fashion and try with several inputs to convince yourself of its working.

    vi. To recognize the integer constant. Token Name is **INT_CONST**

    vii. To recognize the floating-point constant. Token Name is **FLOAT_CONST**

    viii. To recognize comments of the type "// xxxx". Token Name **SINGLE_COMMENT**

    ix. Add rule(s) to recognize comments of type /* xxxx */. Token name **MULTI_COMMENT.**



```
    return new Token(null);
%eofval}
%%
"//".* {System.out.print("<SINGLE_COMMENT"+">");}
"/*".*"*/" {System.out.print("<MULTI_COMMENT"+">");}
"if" {System.out.print("<IF"+">");}
"for" {System.out.print("<FOR"+">");}
"while" {System.out.print("<WHILE"+">");}
"int" {System.out.print("<INT"+">");}
"return" {System.out.print("<RETURN"+">");}
({letter})({letter}|{digit})* {System.out.print("<ID,"+yytext()+
{digit}+ {System.out.print("<INT_CONST,"+yytext()+">");}
{digit}+(.){digit}+ {System.out.print("<FLOAT_CONST,"+yytext()+'
{whitespace}+ {System.out.print(yytext());}
(\{) {System.out.print("<OPEN_PAREN,"+yytext()+">");}
(\}) {System.out.print("<CLOSE_PAREN,"+yytext()+">");}
. {System.out.print("<SYMBOL>");}
```

```
int fun(int n){
    //initialise return variable
    int ans=1;
    for(int i=2;i<n;i++){
        ans*=i;
    }
    if
    /*some comments*/
    return ans;
}
```

```
(base) nspk@Crisp lab4 % jflex q3.jlex
Reading "q3.jlex"

Warning : Macro "special" has been declared but never used.
Constructing NFA : 73 states in NFA
Converting NFA to DFA :
.....................................
39 states before minimization, 32 states in minimized DFA
Old file "Yylex.java" saved as "Yylex.java~"
Writing code to "Yylex.java"
(base) nspk@Crisp lab4 % javac Yylex.java
(base) nspk@Crisp lab4 % java Main
<INT> <ID,fun><SYMBOL><INT> <ID,n><SYMBOL><OPEN_PAREN,{>
    <SINGLE_COMMENT>
    <INT> <ID,ans><SYMBOL><INT_CONST,1><SYMBOL>
    <FOR><SYMBOL><INT> <ID,i><SYMBOL><INT_CONST,2><SYMBOL><ID,i><SYMBOL><ID,n><SYMBOL><ID,i><SYMBOL><SYMBOL><SYMBOL><OPEN_PAREN,{>
        <ID,ans><SYMBOL><SYMBOL><ID,i><SYMBOL>
    <CLOSE_PAREN,}>
    <IF>
    <MULTI_COMMENT>
    <RETURN> <ID,ans><SYMBOL>
<CLOSE_PAREN,}>%
(base) nspk@Crisp lab4 %
```