LAB SHEET 4-INTRODUCTION TO JLEX

1. Try the following JLex Program to recognize a 5 letter word which starts with P/p and ends with T/t.
2. Understand the program by trying out with different inputs and by tweaking the code and seeing the effect and also by going through the generated Java code

Don't copy the code from the PDF as it may generate many errors in Jlex.

```
import java.io.*;
class Main{
public static void main(String args[]) throws IOException{
Yylex lex=new Yylex(System.in);
Token token=lex.yylex();
while(token.text != null ) {
token= lex.yylex();
}
}
}
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);
%eofval}
%%
[Pp]{letter}{letter}{letter}[Tt]{ System.out.print("<Letter starting with P or p and  ending
with T or t," + yytext() + ">"); }
{whitespace}+ {/*Skip white spaces*/}
```

Here is how to generate code and run it on input:

```
CMD> jlex Yylex ==> Generates Yylex.java
CMD> javac Yylex.java ==> Generates Yylex.class, Main.class, Token.class
CMD> java Main    //  Type strings at the terminal and ctrl-D to exit
```

Example: Peryt

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

```
import java.io.*;
class Main {
public static void main(String args[]) throws IOException {
Yylex lex = new Yylex(System.in);
Token token = lex.yylex();
while(token.text != null ) {
token = lex.yylex();
}
}
}
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);
%eofval}
%%
{letter}({letter}|{digit})*{System.out.print("<A valid Identifier,"+yytext()+">");}
{whitespace}+ {/*Skip white spaces*/}
```

4. Write JLex code for the following:
   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.
   ii. To recognize any Java identifier that does not end with an underscore.
   iii. To recognize the keyword "if" in addition to identifiers. (Place the rule of “if” above the rule of identifier.)
   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, viz.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 comments of the type "// xxxx".
vii. Add rule(s) to recognize comments of type /* xxxx */.
viii. All strings that start with P and end with !.
ix. All strings that start with a number and end with an alphabet.