Lab Sheet 4

JLex Assignments

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

1. Create a file **Yylex** and type in the following code**. Don't COPY and PASTE, it will result in error.**

```
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 ) {
        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] { return new Token(yytext() ) ;  }
{whitespace}+ {/*Skip white spaces*/}
.                    { }
```

2. Steps to execute and run a Jlex file

   a. **cmd**> jlex Yylex ➔ *This command will generate the file Yylex.java*
   b. **cmd**> javac Yylex.java ➔ *Compile the file which generates Yylex.class, Main.class, Token.class*
   c. **cmd**> java Main *// Type strings at the terminal and ctrl-D to exit*
   Peryt
   Letter starting with P or p and ending with T or t

B. 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*/}
```

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.**