//Nicholas Williams  
import java.io.\*;  
  
public class Lexer {  
 private static final String VALID\_CHARS = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890"  
 + ".,;:<>/\*[]+-=()}{\t ";  
 private static final String LETTERS = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";  
 private static final String NUMBERS = "1234567890";  
 private static final String SYMBOLS = ".,;:<>/\*[]+-=()}{\t \n";  
 private static final int IDENTIFIER\_SIZE = 32;  
  
 public static String currentLine;  
 private static char currentChar;  
 public static int lineNumber = 1;  
 public static int stringIndex = 0;  
 public static boolean error = false;  
 private static boolean Comment = false;  
 private static boolean lookingForToken = true;  
 private static String currentString;  
 private static String tokenValue;  
 private static String tokenType;  
 private static boolean Tokenized = false;  
  
 private static boolean validChar(String s, char x) {  
 if (s.length() == 0)  
 return false;  
 else  
 return s.charAt(0) == x || validChar(s.substring(1), x);  
  
 }  
//checks to see if a character is valid in the language  
 private static void validCharCheck(String args) {  
 int stringLength = args.length();  
 boolean temp = Comment;  
 for (int j = 0; j < stringLength;) {  
 if(!error)  
 {  
 if (temp)  
 {  
 if(args.charAt(j) == '}')  
 {  
 temp = false;  
 }  
 j++;  
 }  
 else if (validChar(VALID\_CHARS, args.charAt(j))) {  
 if(args.charAt(j) == '{')  
 {  
 temp = true;  
 }  
 j++;  
 } else {  
 System.out.println("??? Lexer error: Bad character at line " + (lineNumber) + ", character " + (j+1));  
 error = true;  
 j++;  
 }  
 }  
 else j++;  
 }  
 }  
//checks to see if the character after a decimal is valid when part of a number  
 private static void decimalCheck(int length) {  
 if (stringIndex + 1 < length) {  
 if (validChar(NUMBERS, currentLine.charAt(stringIndex + 1))) {  
 currentString = currentString + currentChar;  
 stringIndex++;  
 currentChar = currentLine.charAt(stringIndex);  
 } else {  
 lookingForToken = false;  
 }  
 } else {  
 lookingForToken = false;  
 }  
 }  
//checks to see if the current string is an exponential number  
 private static void exponentialCheck(int length) {  
 if (currentChar == 'E' || currentChar == 'e')   
 {  
 if (stringIndex + 1 >= length) {  
 lookingForToken = false;  
 error = true;  
 System.out.println("??? Lexer error: Bad character at line " + lineNumber + ", character " + (stringIndex+1));  
 Tokenized = false;  
 } else if (validChar(LETTERS, currentLine.charAt(stringIndex + 1))) {  
 lookingForToken = false;  
 error = true;  
 System.out.println("??? Lexer error: Bad character at line " + lineNumber + ", character " + (stringIndex+2));  
 Tokenized = false;  
 } else if (validChar(NUMBERS, currentLine.charAt(stringIndex + 1))) {  
 currentString = currentString + Character.toUpperCase(currentChar);  
 stringIndex++;  
  
 } else {  
 if (stringIndex + 2 < length)   
 {  
 if ((currentLine.charAt(stringIndex + 1) == '-') || currentLine.charAt(stringIndex + 1) == '+') {  
 if (validChar(NUMBERS, currentLine.charAt(stringIndex + 2))) {  
 currentString = currentString + Character.toUpperCase(currentChar) + currentLine.charAt(stringIndex + 1);  
 stringIndex++;  
 stringIndex++;  
 } else{  
 lookingForToken = false;  
 error = true;  
 Tokenized = false;  
 System.out.println("??? Lexer error: Bad character at line " + lineNumber + ", character " + (stringIndex+3));  
 }  
 } else  
 {  
 lookingForToken = false;  
 error = true;  
 System.out.println("??? Lexer error: Bad character at line " + lineNumber + ", character " + (stringIndex+2));  
 }  
 } else{  
 lookingForToken = false;  
 error = true;  
 Tokenized = false;  
 System.out.println("??? Lexer error: Bad character at line " + lineNumber + ", character " + (stringIndex+1));  
 }  
 }  
 }  
  
 else {  
 error = true;  
 System.out.println("??? Lexer error: Bad character at line " + lineNumber + ", character " + (stringIndex+1));  
 }  
 }  
  
 //gets the next token.   
 public static String[] GetNextToken() {  
 Tokenized = false;  
 lookingForToken = true;  
 //if end of file  
 if (stringIndex == currentLine.length())   
 {  
 if (Comment)   
 {  
 System.out.println("??? Lexer error: Bad character at line " + lineNumber + ", character " + (stringIndex+1));  
 error = true;  
 lookingForToken = false;  
 Tokenized = false;  
 }  
 else  
 {  
 tokenValue = "~";  
 tokenType = "ENDOFFILE";  
 Tokenized = true;  
 stringIndex++;  
 }  
 } else {  
 int length = currentLine.length();  
 while ((stringIndex < length) && (lookingForToken))   
 {  
 currentChar = currentLine.charAt(stringIndex);  
 if (Comment == true)   
 {  
 if (currentChar == '}') {  
 Comment = false;  
 stringIndex++;  
 } else {  
 stringIndex++;  
 }  
 }   
 else   
 {  
 if (currentChar == '{')   
 {  
 Comment = true;  
 Tokenized = false;  
 stringIndex++;  
 }   
 else if (currentChar == '}')   
 {  
 currentString = ("??? Lexer error: Bad character at line " + lineNumber + ", character " + (stringIndex+1));  
 System.out.println(currentString);  
 Tokenized = false;  
 error = true;  
 lookingForToken = false;  
 }   
 // identifiers and key words  
 else if (validChar(LETTERS, currentChar))   
 { Tokenized = true;  
 currentString = "";  
 while ((stringIndex < length) && lookingForToken)   
 {  
 currentChar = currentLine.charAt(stringIndex);  
 if (!validChar(SYMBOLS, currentChar))   
 {  
 currentString = currentString + Character.toUpperCase(currentChar);  
 stringIndex++;  
 }   
 else  
 lookingForToken = false;  
 }  
 lookingForToken = false;  
 if (currentString.equals("IF"))  
 {  
 tokenValue = ("~");  
 tokenType = "IF";  
 }  
 else if (currentString.equals("PROGRAM"))  
 {  
 tokenValue = ("~");  
 tokenType = "PROGRAM";  
 }  
 else if (currentString.equals("RESULT"))  
 {  
 tokenValue = ("~");  
 tokenType = "RESULT";  
 }  
 else if (currentString.equals("BEGIN"))  
 {  
 tokenValue = ("~");  
 tokenType = "BEGIN";  
 }  
 else if (currentString.equals("INTEGER"))  
 {  
 tokenValue = ("~");  
 tokenType = "INTEGER";  
 }  
 else if (currentString.equals("THEN"))  
 {  
 tokenValue = ("~");  
 tokenType = "THEN";  
 }  
 else if (currentString.equals("END"))  
 {  
 tokenValue = ("~");  
 tokenType = "END";  
 }  
 else if (currentString.equals("REAL"))  
 {  
 tokenValue = ("~");  
 tokenType = "REAL";  
 }  
 else if (currentString.equals("ELSE"))  
 {  
 tokenValue = ("~");  
 tokenType = "ELSE";  
 }  
 else if (currentString.equals("VAR"))  
 {  
 tokenValue = ("~");  
 tokenType = "VAR";  
 }  
 else if (currentString.equals("ARRAY"))  
 {  
 tokenValue = ("~");  
 tokenType = "ARRAY";  
 }  
 else if (currentString.equals("WHILE"))  
 {  
 tokenValue = ("~");  
 tokenType = "WHILE";  
 }  
 else if (currentString.equals("FUNCTION"))  
 {  
 tokenValue = ("~");  
 tokenType = "FUNCTION";  
 }  
 else if (currentString.equals("OF"))  
 {  
 tokenValue = ("~");  
 tokenType = "OF";  
 }  
 else if (currentString.equals("DO"))  
 {  
 tokenValue = ("~");  
 tokenType = "DO";  
 }  
 else if (currentString.equals("PROCEDURE"))  
 {  
 tokenValue = ("~");  
 tokenType = "PROCEDURE";  
 }  
 else if (currentString.equals("NOT"))  
 {  
 tokenValue = ("~");  
 tokenType = "NOT";  
 }  
 else if (currentString.equals("DIV"))  
 {  
 tokenValue = ("3");  
 tokenType = "MULOP";  
 }  
 else if (currentString.equals("MOD"))  
 {  
 tokenValue = ("4");  
 tokenType = "MULOP";  
 }  
 else if (currentString.equals("AND"))  
 {  
 tokenValue = ("5");  
 tokenType = "MULOP";  
 }  
 else if (currentString.equals("OR"))  
 {  
 tokenValue = ("3");  
 tokenType = "ADDOP";  
 }  
 else   
 {  
 if (currentString.length() <= IDENTIFIER\_SIZE)   
 {  
 tokenValue = (currentString);  
 tokenType = "IDENTIFIER";  
 }   
 else   
 {  
 System.out.println("??? Lexor error: Identifier too long at line " + lineNumber);  
 error = true;  
 Tokenized = false;  
 }  
 }  
 }  
 // numbers  
 else if (validChar(NUMBERS, currentChar))  
 {  
 Tokenized = true;  
 boolean exponential = false;  
 boolean decimal = false;  
 boolean while1 = true;  
 boolean while2 = true;  
 boolean while3 = true;  
 currentString = "" + currentChar;  
 stringIndex++;  
 while ((stringIndex < length) && (!error) && while1) {  
 currentChar = currentLine.charAt(stringIndex);  
 if (validChar(NUMBERS, currentChar)) {  
 currentString = currentString + currentChar;  
 stringIndex++;  
 } else  
 while1 = false;  
 }  
 if (stringIndex < length) {  
 if (validChar(LETTERS, currentChar)) {  
 exponentialCheck(length);  
 exponential = true;  
 } else if ((currentChar == '.')) {  
 decimalCheck(length);  
 if(lookingForToken) decimal = true;  
 } else  
 lookingForToken = false;  
 }  
 if (lookingForToken) {  
 while ((stringIndex < length) && (!error) && while2) {  
 currentChar = currentLine.charAt(stringIndex);  
 if (validChar(NUMBERS, currentChar)) {  
 currentString = currentString + currentChar;  
 stringIndex++;  
 } else  
 while2 = false;  
 }  
 if (stringIndex < length) {  
 currentChar = currentLine.charAt(stringIndex);  
 if ((validChar(LETTERS, currentChar)) && (!exponential)) {  
 exponentialCheck(length);  
 exponential = true;  
 } else if (currentChar == '.'){  
 lookingForToken = false;  
 error = true;  
 Tokenized = false;  
 System.out.println("??? Lexer error: Bad character at line " + lineNumber + ", character " + (stringIndex+1));  
 }   
 else if(validChar(LETTERS, currentChar))  
 {  
 lookingForToken = false;  
 error = true;  
 Tokenized = false;  
 System.out.println("??? Lexer error: Bad character at line " + lineNumber + ", character " + (stringIndex+1));  
 }  
 else  
 lookingForToken = false;  
 if (lookingForToken) {  
 while ((stringIndex < length) && (!error) && while3) {  
 currentChar = currentLine.charAt(stringIndex);  
 if (validChar(NUMBERS, currentChar)) {  
 currentString = currentString + currentChar;  
 stringIndex++;  
 } else  
 while3 = false;  
 }  
 if (stringIndex < length) {  
 currentChar = currentLine.charAt(stringIndex);  
 if (currentChar == '.') {  
 lookingForToken = false;  
 error = true;  
 Tokenized = false;  
 System.out.println("??? Lexer error: Bad character at line " + lineNumber + ", character " + (stringIndex+1));  
 }  
   
 else if(validChar(LETTERS, currentChar))  
 {  
 lookingForToken = false;  
 error = true;  
 Tokenized = false;  
 System.out.println("??? Lexer error: Bad character at line " + lineNumber + ", character " + (stringIndex+1));  
 }  
 else  
 lookingForToken = false;  
 }  
 }  
 }  
 }  
   
 if (exponential || decimal) {  
 tokenValue = currentString;  
 tokenType = "REALCONSTANT";  
 } else {  
 tokenValue = currentString;  
 tokenType = "INTCONSTANT";  
 }  
 }   
 else if (validChar(SYMBOLS, currentChar))   
 {   
 if (currentChar == ' ')   
 {  
   
 }  
 else if (currentChar == '\t')  
 {  
   
 }   
 else if (currentChar == '\n')  
 {  
 lineNumber = lineNumber+1;  
 }  
 else if (currentChar == ')')  
 {  
 Tokenized = true;  
 tokenValue = "~";  
 tokenType = "RPAREN";  
 lookingForToken = false;  
 }  
 else if (currentChar == '(')  
 {  
 Tokenized = true;  
 tokenValue = "~";  
 tokenType = "LPAREN";  
 lookingForToken = false;  
 }  
 else if (currentChar == ']')  
 {  
 Tokenized = true;  
 tokenValue = "~";  
 tokenType = "RBRACKET";  
 lookingForToken = false;  
 }  
 else if (currentChar == '[')  
 {  
 Tokenized = true;  
 tokenValue = "~";  
 tokenType = "LBRACKET";  
 lookingForToken = false;  
 }  
 else if (currentChar == ';')  
 {  
 Tokenized = true;  
 tokenValue = "~";  
 tokenType = "SEMICOLON";  
 lookingForToken = false;  
 }  
 else if (currentChar == ',')  
 {  
 Tokenized = true;  
 tokenValue = "~";  
 tokenType = "COMMA";  
 lookingForToken = false;  
 }  
 else if (currentChar == '=')  
 {  
 Tokenized = true;  
 tokenValue = "1";  
 tokenType = "RELOP";  
 lookingForToken = false;  
 }  
 else if (currentChar == ':')  
 {  
 Tokenized = true;  
 stringIndex++;  
 if(stringIndex < length)  
 {  
 if (currentLine.charAt(stringIndex) == '=')  
 {  
 tokenValue = "~";  
 tokenType = "ASSIGNOP";  
 lookingForToken = false;  
 }  
 else  
 {  
 tokenValue = "~";  
 tokenType = "COLON";  
 lookingForToken = false;  
 stringIndex--;  
 }  
 }  
 else  
 {  
 tokenValue = "~";  
 tokenType = "COLON";  
 lookingForToken = false;  
 }  
 }  
 else if (currentChar == '.')  
 {  
 Tokenized = true;  
 stringIndex++;  
 if(stringIndex < length)  
 {  
 if (currentLine.charAt(stringIndex) == '.')  
 {  
 tokenValue = "~";  
 tokenType = "DOUBLEDOT";  
 lookingForToken = false;  
 }  
 else  
 {  
 tokenValue = "~";  
 tokenType = "ENDMARKER";  
 lookingForToken = false;  
 stringIndex--;  
 }  
 }  
 else  
 {  
 tokenType = "ENDMARKER";  
 tokenValue = "~";  
 lookingForToken = false;  
 stringIndex--;  
 }  
 }  
 else if (currentChar == '\*')  
 {  
 Tokenized = true;  
 tokenValue = "1";  
 tokenType = "MULOP";  
 lookingForToken = false;  
 }  
 else if (currentChar == '/')  
 {  
 Tokenized = true;  
 tokenValue = "2";  
 tokenType = "MULOP";  
 lookingForToken = false;  
 }  
 else if (currentChar == '-')  
 {  
 Tokenized = true;  
 if(tokenType.equals("RPAREN")||tokenType.equals("LPAREN")||tokenType.equals("REALCONSTANT")||tokenType.equals("IDENTIFIER")||tokenType.equals("INTCONSTANT"))  
 {  
 tokenValue = "2";  
 tokenType = "ADDOP";  
 lookingForToken = false;  
 }  
 else  
 {  
 tokenValue = "~";  
 tokenType = "UMINUS";  
 lookingForToken = false;  
 }  
 }  
 else if (currentChar == '+')  
 {  
 Tokenized = true;  
 if(tokenType.equals("RPAREN")||tokenType.equals("LPAREN")||tokenType.equals("REALCONSTANT")||tokenType.equals("IDENTIFIER")||tokenType.equals("INTCONSTANT"))  
 {  
 tokenValue = "1";  
 tokenType = "ADDOP";  
 lookingForToken = false;  
 }  
 else  
 {  
 tokenValue = "~";  
 tokenType = "UPLUS";  
 lookingForToken = false;  
 }  
 }  
 else if (currentChar == '<')  
 {  
 Tokenized = true;  
 stringIndex++;  
 if (stringIndex < length)  
 {  
 if (currentLine.charAt(stringIndex) == '>')  
 {  
 tokenValue = "2";  
 tokenType = "RELOP";  
 lookingForToken = false;  
 }  
 else if (currentLine.charAt(stringIndex) == '=')  
 {  
 tokenValue = "5";  
 tokenType = "RELOP";  
 lookingForToken = false;   
 }  
 else  
 {  
 tokenValue = "3";  
 tokenType = "RELOP";  
 lookingForToken = false;  
 stringIndex--;  
 }  
 }  
 else  
 {  
 tokenValue = "3";  
 tokenType = "RELOP";  
 lookingForToken = false;  
 }  
 }  
 else if (currentChar == '>')  
 {  
 Tokenized = true;  
 stringIndex++;  
 if (stringIndex < length)  
 {  
 if (currentLine.charAt(stringIndex) == '=')  
 {  
 tokenValue = "6";  
 tokenType = "RELOP";  
 lookingForToken = false;  
 }  
 else  
 {  
 tokenValue = "4";  
 tokenType = "RELOP";  
 lookingForToken = false;  
 stringIndex--;  
 }  
 }  
 else  
 {  
 tokenValue = "4";  
 tokenType = "RELOP";  
 lookingForToken = false;  
 }  
 }  
 stringIndex++;  
 }  
 }  
 }  
 }  
 if (Tokenized){  
 //System.out.println(tokenType + ", " + tokenValue);  
 }else{  
 tokenValue = "~";  
 tokenType = "ENDOFFILE";  
 }  
 String [] token = new String[2];  
 token[0] = tokenType;  
 token[1] = tokenValue;  
 return token;  
 }  
  
 public static void load(String fileName) throws FileNotFoundException, IOException{  
 File file = new File(fileName);  
 String st;  
 currentLine = "";  
 BufferedReader br = new BufferedReader(new FileReader(file));  
 while (((st = br.readLine()) != null) && (!error)) {  
 stringIndex = 0;  
 validCharCheck(st);  
 currentLine = currentLine + '\n' + st;  
 lineNumber++;  
 }  
 lineNumber = 1;  
 br.close();  
 }  
  
 public static void main(String args[]) throws FileNotFoundException, IOException   
 {  
   
 load("test.txt");  
 while ((stringIndex < currentLine.length()) && (!error))   
 {  
 GetNextToken();  
 // System.out.println(tokenType + tokenValue);  
 // System.out.println(lineNumber);  
 lookingForToken = true;  
 }  
   
 if (!error)  
 GetNextToken();  
 }  
}