LAPORAN TUGAS BESAR
TEORI BAHASA DAN AUTOMATA

Disusun oleh:

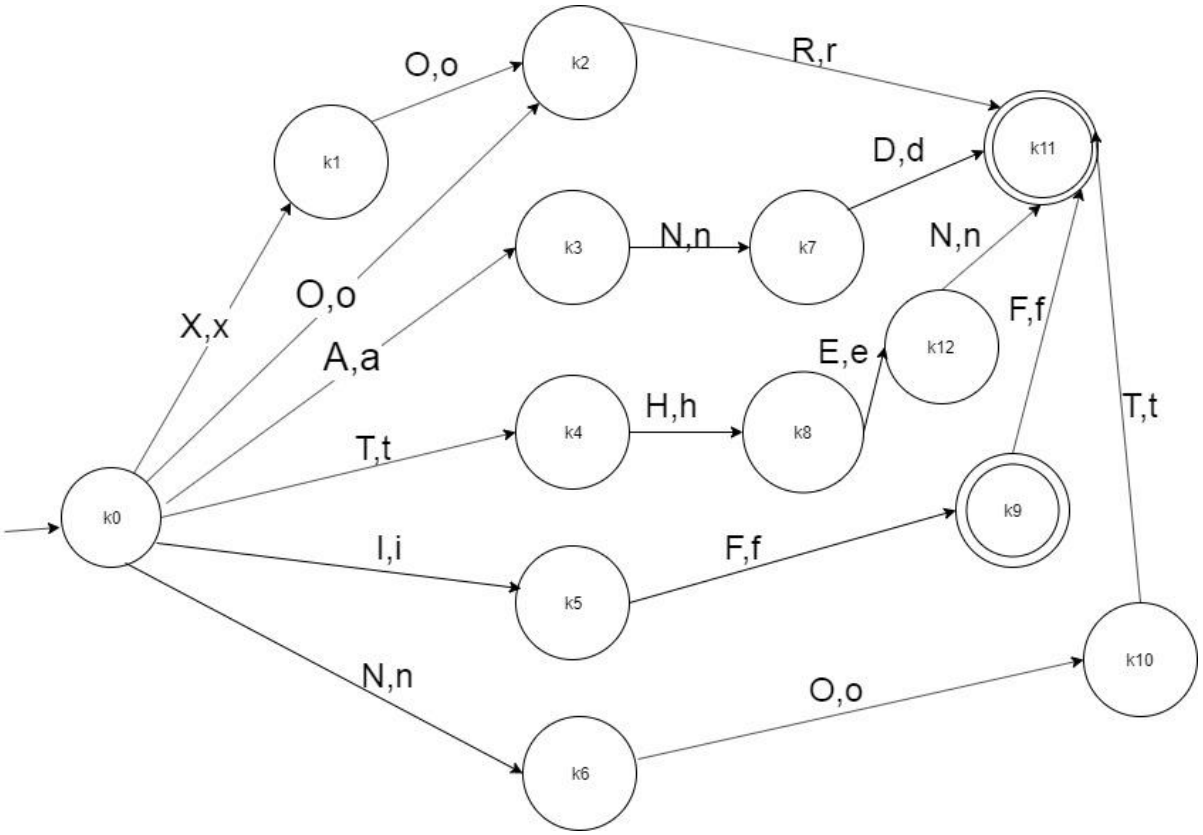
Revi Chandra Riana	**1301198512**
Muhammad Faisal Amir	**1301198497**
Hildan Fawwaz Naufal	**1301198514**
Monica Liviandra	**1301198517**

TELKOM UNIVERSITY
BANDUNG
2019

A.Rancangan Automata

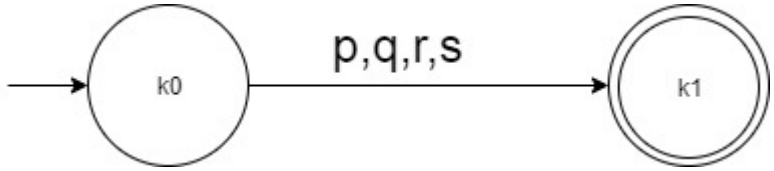
Finite Automata

Automata yang digunakan untuk mengecek apakah sebuah string termasuk Operator



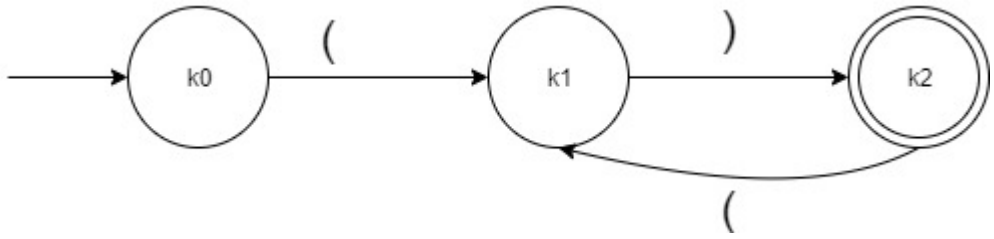
state	a	d	e	f	h	i	n	o	r	t	x
k0	k3	error	error	error	error	k5	k6	k2	error	k4	k1
k1	error	error	error	error	error	error	error	k2	error	error	error
k2	error	error	error	error	error	error	error	error	k11	error	error
k3	error	error	error	error	error	error	k7	error	error	error	error
k4	error	error	error	error	k8	error	error	error	error	error	error
k5	error	error	error	k9	error	error	error	error	error	error	error
k6	error	error	error	error	error	error	error	k10	error	error	error
k7	error	k11	error	error	error	error	error	error	error	error	error
k8	error	error	k12	error	error	error	error	error	error	error	error
k9	error	error	error	k11	error	error	error	error	error	error	error
k10	error	error	error	error	error	error	error	error	error	k11	error
k11	error	error	error	error	error	error	error	error	error	error	error
k12	error	error	error	error	error	error	k11	error	error	error	error

Automata yang digunakan untuk mengecek apakah sebuah string termasuk Operand



state	p	q	r	s
k0	k1	k1	k1	k1
k1	error	error	error	error

Automata yang digunakan untuk mengecek apakah sebuah string termasuk Grouping



state	(	)
k0	k1	error
k1	error	k2
k2	error	error

B. Context Free Grammar

CFG (Context Free Grammar) yang digunakan untuk menyusun logika proposisi yang valid dari inputan *user* adalah sebagai berikut.

S → A | BA | BK | KCK | KDK | KEK |KHK | SCS | SDS | SES | SHS | FSGS | FKGK | FAGA | FAGK | ADK | AEK | ACK | AHK

A → p,q,r,s

B → not

C → and

D → or

E → xor

F → if

G → then

H → iff

I → (

J →)

K → ISJ

C.Source Code Program

Main Class

```
private void teamView(){
    System.out.println(TEXT_TITLE_PROGRAM);
    System.out.println(VIEW_LINE);
    System.out.println("Muhammad Faisal Amir (1301198
    System.out.println("Hildan Fawwaz        (1301198
    System.out.println("Monica liviandra     (1301198
    System.out.println("Revi Chandra Riana   (1301198
    System.out.println("IFX - 43 - 02");
```

Method diatas digunakan untuk menampilkan nama kelompok kami dan kelas.

```
private void introView(){
    System.out.println("List Token : ");
    System.out.println(VIEW_LINE);
    System.out.println("TOKEN_PROPOSITION \t: "  + TOKEN_PROPOSITION);
    System.out.println("TOKEN_NOT \t\t\t: "        + TOKEN_NOT);
    System.out.println("TOKEN_AND \t\t\t: "        + TOKEN_AND);
    System.out.println("TOKEN_OR \t\t\t: "         + TOKEN_OR);
    System.out.println("TOKEN_XOR \t\t\t: "        + TOKEN_XOR);
    System.out.println("TOKEN_IF \t\t\t: "         + TOKEN_IF);
    System.out.println("TOKEN_THEN \t\t\t: "       + TOKEN_THEN);
    System.out.println("TOKEN_IFF \t\t\t: "        + TOKEN_IFF);
    System.out.println("TOKEN_( \t\t\t: "          + TOKEN_BRACKET_OPEN);
    System.out.println("TOKEN_) \t\t\t: "      + TOKEN_BRACKET_CLOSE);
    System.out.println(VIEW_LINE);
}
```

Method diatas digunakan untuk menampilkan list token yang ada dan perubahannya dalam bentuk angka

```
private void onCreate(){
    // Call all function here
    runLexicalAnalyzer();
}

private void runLexicalAnalyzer(){
    System.out.print(TEXT_INPUT);
    Scanner scanner = new Scanner(System.in);
    String input = scanner.nextLine();
    new Logic().parseString(input);
}
```

Method diatas digunakan untuk memanggil logika running program dari class Logic, yang kemudian method onCreate nanti akan dipanggil lagi oleh main method untuk menjalankan program.

```
public static void main(String[] args) {
    // Write your code here
    new Main().teamView();
    new Main().introView();
    new Main().onCreate();
}
```

Method diatas digunakan untuk memanggil method teamView,introView,onCreate yang berfungsi untuk menyatukan dan menstruktur urutan pemanggilan method agar program ketika dijalankan akan sesuai dengan urutan.

Logic Class

```
public class Logic {

    private StringChecker stringChecker = new StringChecker();
```

Kode diatas digunakan inisialisasi objek StringChecker.

```
public int getParseToken(char[] arrayString) {
    if (stringChecker.checkStringProposition(arrayString)) {
        return TOKEN_PROPOSITION;
    } else if (stringChecker.checkStringNot(arrayString)) {
        return TOKEN_NOT;
    } else if (stringChecker.checkStringAnd(arrayString)) {
        return TOKEN_AND;
    } else if (stringChecker.checkStringOr(arrayString)) {
        return TOKEN_OR;
    } else if (stringChecker.checkStringXor(arrayString)) {
        return TOKEN_XOR;
    } else if (stringChecker.checkStringIf(arrayString)) {
        return TOKEN_IF;
    } else if (stringChecker.checkStringThen(arrayString)) {
        return TOKEN_THEN;
    } else if (stringChecker.checkStringIff(arrayString)) {
        return TOKEN_IFF;
    } else if (stringChecker.checkStringBracketOpen(arrayString)) {
        return TOKEN_BRACKET_OPEN;
    } else if (stringChecker.checkStringBracketClose(arrayString)) {
        return TOKEN_BRACKET_CLOSE;
    } else {
        return 0;
    }
}
```

Method diatas digunakan untuk mengembalikan nilai token masing-masing dari inputan program berupa arrayCharacter.

```
private String[] separatorSentence(String sentence) { return sentence.split(" "); }

private char[] separatorString(String splitString) { return splitString.toCharArray(); }

private ArrayList<String> arraySplitString(String input) {
    ArrayList<String> partTemp = new ArrayList<>();
    Collections.addAll(partTemp, separatorSentence(input));
    return partTemp;
}

private boolean checkValidPropotionalLogic(ArrayList<Integer> integerArrayList) {
    return (integerArrayList.get(0) == 2) && (integerArrayList.get(1) == 1);
}

private boolean checkValidCloseBracked(ArrayList<Integer> arrayListToken, int position) {
    return (arrayListToken.get(position) == 10) && (arrayListToken.get(position + 1) == 10);
}

private boolean checkValidOpenBracked(ArrayList<Integer> arrayListToken, int position) {
    return (arrayListToken.get(position) == 9) && (arrayListToken.get(position + 1) == 9);
}

private boolean checkValidBracket(ArrayList<Integer> arrayListToken, int position) {
    return checkValidOpenBracked(arrayListToken, position) || checkValidCloseBracked(arrayListToken, position);
}
```

Beberapa method diatas, dimulai dari yang paling atas digunakan untuk memisahkan inputan yang dipisahkan oleh spasi. Kemudian method selanjutnya, pengecekan validasi dari proposisi, kurung tutup yang menumpuk, buka kurung yang menumpuk.

```
private void validatorLexicalParse(ArrayList<Integer> arrayListToken) {
    String checkValid = TEXT_VALID;
    if ((arrayListToken.size() == 1) && (arrayListToken.get(0) <= 10)) {
        checkValid = TEXT_NOT_VALID;
        System.out.print(arrayListToken.get(0));
    } else if ((arrayListToken.size() == 2)) {
        if (!checkValidPropotionalLogic(arrayListToken)) {
            checkValid = TEXT_NOT_VALID;
        }
        for (Integer integer : arrayListToken) {
            System.out.print(integer + "\t");
        }
    } else {
        for (int i = 0; i < arrayListToken.size(); i++) {
            if (arrayListToken.get(i) > 0) {
                System.out.print(arrayListToken.get(i) + "\t");
                if (i != arrayListToken.size() - 1) {
                    if (arrayListToken.get(i) == arrayListToken.get(i + 1)) {
                        if (!checkValidBracket(arrayListToken, i)) {
                            checkValid = TEXT_NOT_VALID;
                        }
                    }
                }
            } else {
                System.out.print(TOKEN_ERROR + " ");
                checkValid = TEXT_NOT_VALID;
                break;
            }
        }
    }

    System.out.println();
    System.out.println(TEXT_RESULT + checkValid);
}
```

Method diatas digunakan untuk pengecekan valid ato tidaknya dari inputan token apakah sesuai dengan aturan penulisan logika proposisi ato tidak.

```
public void parseString(String input) {
    ArrayList<Integer> arrayListToken = new ArrayList<>();
    ArrayList<String> arrayListWord = arraySplitString(input);
    for (String stringWord : arrayListWord) {
        char[] word = separatorString(stringWord);
        arrayListToken.add(getParseToken(word));
    }
    System.out.print(TEXT_OUTPUT);
    validatorLexicalParse(arrayListToken);
}
```

Method diatas digunakan untuk mengganti string menjadi token hasil inputan berupa string.

Constant Class

```
public class Constant {

    public static final String TEXT_TITLE_PROGRAM = "Lexical Analyzer";
    // ------------------------------------------------------------------------------------------------------------------------
    public static final String TEXT_INPUT  = "Input (String)        : ";
    public static final String TEXT_OUTPUT = "Output (Token Lexic)  : ";
    public static final String TEXT_RESULT = "Output                : ";
    public static final String TEXT_VALID = "Valid";
    public static final String TEXT_NOT_VALID = "Tidak Valid";
    public static final String VIEW_LINE = "-----------------------------------------------";
    // ------------------------------------------------------------------------------------------------------------------------
    public static final int TOKEN = 0;
    public static final int TOKEN_PROPOSITION = 1;
    public static final int TOKEN_NOT = 2;
    public static final int TOKEN_AND = 3;
    public static final int TOKEN_OR = 4;
    public static final int TOKEN_XOR = 5;
    public static final int TOKEN_IF = 6;
    public static final int TOKEN_THEN = 7;
    public static final int TOKEN_IFF = 8;
    public static final int TOKEN_BRACKET_OPEN = 9; // (
    public static final int TOKEN_BRACKET_CLOSE = 10; // )
    public static final String TOKEN_ERROR = "error";
    // ------------------------------------------------------------------------------------------------------------------------

}
```

Class diatas digunakan untuk inisialisasi constanta token.

CharChecker Class

```java
public class CharChecker {

    // List String Lexic : p, q, r, s, not, and, or, xor, if, then, iff, (, )
    // List Character : a, d, e, f, h, i, n, o, p, q, r, s, t, x, (, )

    boolean checkA(char input) { return (input == 'a') || input == 'A'; }

    boolean checkD(char input) { return (input == 'd') || input == 'D'; }

    boolean checkE(char input) { return (input == 'e') || input == 'E'; }

    boolean checkF(char input) { return (input == 'f') || input == 'F'; }

    boolean checkH(char input) { return (input == 'h') || input == 'H'; }

    boolean checkI(char input) { return (input == 'i') || input == 'I'; }

    boolean checkN(char input) { return (input == 'n') || input == 'N'; }

    boolean checkO(char input) { return (input == 'o') || input == 'O'; }

    boolean checkP(char input) { return (input == 'p') || input == 'P'; }

    boolean checkQ(char input) { return (input == 'q') || input == 'Q'; }

    boolean checkR(char input) { return (input == 'r') || input == 'R'; }

    boolean checkS(char input) { return (input == 's') || input == 'S'; }

    boolean checkT(char input) { return (input == 't') || input == 'T'; }

    boolean checkX(char input) { return (input == 'x') || input == 'X'; }

    boolean checkBracketOpen(char input) { return input == '('; }

    boolean checkBracketClose(char input) { return input == ')'; }


    // Check length for proposition p,q,r,s,(,)
    boolean checkSizeOneChar(char[] arrayString) { return arrayString.length == 1; }

    // Check length for if,or,
    boolean checkSizeTwoChar(char[] arrayString) { return arrayString.length == 2; }

    // Check length for not,and,xor,iff
    boolean checkSizeThreeChar(char[] arrayString) { return arrayString.length == 3; }

    // Check length for then
    boolean checkSizeFourChar(char[] arrayString) { return arrayString.length == 4; }
```

Class diatas digunakan untuk pengecekan karakter yang diterima oleh program.

StringChecker Class

```
public class StringChecker {

    // List String Lexic : p, q, r, s, not, and, or, xor, if, then, iff, (, )

    private CharChecker charChecker = new CharChecker();
```

Untuk menginisialisasi objek CharChecker

```
    private boolean checkStringP(char[] arrayString){
        if (charChecker.checkSizeOneChar(arrayString)){
            return charChecker.checkP(arrayString[0]);
        } else {
            return false;
        }
    }

    private boolean checkStringQ(char[] arrayString){
        if (charChecker.checkSizeOneChar(arrayString)){
            return charChecker.checkQ(arrayString[0]);
        } else {
            return false;
        }
    }

    private boolean checkStringR(char[] arrayString){
        if (charChecker.checkSizeOneChar(arrayString)){
            return charChecker.checkR(arrayString[0]);
        } else {
            return false;
        }
    }

    private boolean checkStringS(char[] arrayString){
        if (charChecker.checkSizeOneChar(arrayString)){
            return charChecker.checkS(arrayString[0]);
        } else {
            return false;
        }
    }

    public boolean checkStringProposition(char[] arrayString){
        return checkStringP(arrayString)
                || checkStringQ(arrayString)
                || checkStringR(arrayString)
                || checkStringS(arrayString);
    }
```

Gabungan method diatas digunakan untuk mengecek apakah proposisi p,q,r,s atau bukan.

```
public boolean checkStringBracketOpen(char[] arrayString){
    if (charChecker.checkSizeOneChar(arrayString)){
        return charChecker.checkBracketOpen(arrayString[0]);
    } else {
        return false;
    }
}

public boolean checkStringBracketClose(char[] arrayString){
    if (charChecker.checkSizeOneChar(arrayString)){
        return charChecker.checkBracketClose(arrayString[0]);
    } else {
        return false;
    }
}
```

Gabungan method diatas digunakan untuk apakah karakter yang sedang dicek sekarang, adalah buka kurung atau tutup kurung.

```
public boolean checkStringIf(char[] arrayString) {
    if (charChecker.checkSizeTwoChar(arrayString)) {
        if (charChecker.checkI(arrayString[0])) {
            return charChecker.checkF(arrayString[1]);
        } else {
            return false;
        }
    } else {
        return false;
    }
}

public boolean checkStringOr(char[] arrayString) {
    if (charChecker.checkSizeTwoChar(arrayString)) {
        if (charChecker.checkO(arrayString[0])) {
            return charChecker.checkR(arrayString[1]);
        } else {
            return false;
        }
    } else {
        return false;
    }
}

public boolean checkStringAnd(char[] arrayString) {
    if (charChecker.checkSizeThreeChar(arrayString)) {
        if (charChecker.checkA(arrayString[0])) {
            if (charChecker.checkN(arrayString[1])) {
                return charChecker.checkD(arrayString[2]);
            } else {
                return false;
            }
        } else {
            return false;
        }
    } else {
        return false;
    }
}
```

Gabungan method diatas digunakan untuk mengecek beberapa karakter sekarang adalah termasuk operator if atau or atau and.

```
public boolean checkStringNot(char[] arrayString) {
    if (charChecker.checkSizeThreeChar(arrayString)) {
        if (charChecker.checkN(arrayString[0])) {
            if (charChecker.checkO(arrayString[1])) {
                return charChecker.checkT(arrayString[2]);
            } else {
                return false;
            }
        } else {
            return false;
        }
    } else {
        return false;
    }
}

public boolean checkStringXor(char[] arrayString) {
    if (charChecker.checkSizeThreeChar(arrayString)) {
        if (charChecker.checkX(arrayString[0])) {
            if (charChecker.checkO(arrayString[1])) {
                return charChecker.checkR(arrayString[2]);
            } else {
                return false;
            }
        } else {
            return false;
        }
    } else {
        return false;
    }
}
```

Gabungan method diatas digunakan untuk mengecek beberapa karakter sekarang adalah termasuk operator not atau Xor.

```
public boolean checkStringIff(char[] arrayString) {
    if (charChecker.checkSizeThreeChar(arrayString)) {
        if (charChecker.checkI(arrayString[0])) {
            if (charChecker.checkF(arrayString[1])) {
                return charChecker.checkF(arrayString[2]);
            } else {
                return false;
            }
        } else {
            return false;
        }
    } else {
        return false;
    }
}

public boolean checkStringThen(char[] arrayString) {
    if (charChecker.checkSizeFourChar(arrayString)) {
        if (charChecker.checkT(arrayString[0])) {
            if (charChecker.checkH(arrayString[1])) {
                if (charChecker.checkE(arrayString[2])) {
                    return charChecker.checkN(arrayString[3]);
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } else {
            return false;
        }
    } else {
        return false;
    }
}
```

Gabungan method diatas digunakan untuk mengecek beberapa karakter sekarang adalah termasuk operator iff atau then.

Hasil Program

Saat program dijalankan maka program akan meminta inputan berupa logika proposisi

```
"C:\Program Files (x86)\Java\jdk1.8.0_102\bin\java" ...
Lexical Analyzer
-----------------------------------------------
Muhammad Faisal Amir (1301198497)
Hildan Fawwaz        (1301198514)
Monica liviandra     (1301198517)
Revi Chandra Riana   (1301198512)
IFX - 43 - 02
-----------------------------------------------
List Token :
-----------------------------------------------
TOKEN_PROPOSITION   : 1
TOKEN_NOT           : 2
TOKEN_AND           : 3
TOKEN_OR            : 4
TOKEN_XOR           : 5
TOKEN_IF            : 6
TOKEN_THEN          : 7
TOKEN_IFF           : 8
TOKEN_(             : 9
TOKEN_)             : 10
-----------------------------------------------
Input (String)       :
```

Lalu program akan memproses apakah struktur logika proposisi sudah valid atau belum.

Contoh No.1 Valid

```
"C:\Program Files (x86)\Java\jdk1.8.0_102\bin\java" ...
Lexical Analyzer
-----------------------------------------------
Muhammad Faisal Amir (1301198497)
Hildan Fawwaz        (1301198514)
Monica liviandra     (1301198517)
Revi Chandra Riana   (1301198512)
IFX - 43 - 02
-----------------------------------------------
List Token :
-----------------------------------------------
TOKEN_PROPOSITION   : 1
TOKEN_NOT           : 2
TOKEN_AND           : 3
TOKEN_OR            : 4
TOKEN_XOR           : 5
TOKEN_IF            : 6
TOKEN_THEN          : 7
TOKEN_IFF           : 8
TOKEN_(             : 9
TOKEN_)             : 10
-----------------------------------------------
Input (String)        : p and q or r
Output (Token Lexic)  : 1   3   1   4   1
Output                : Valid

Process finished with exit code 0
```

Contoh No.2 Valid

```
"C:\Program Files (x86)\Java\jdk1.8.0_102\bin\java" ...
Lexical Analyzer
-----------------------------------------------
Muhammad Faisal Amir (1301198497)
Hildan Fawwaz        (1301198514)
Monica liviandra     (1301198517)
Revi Chandra Riana   (1301198512)
IFX - 43 - 02
-----------------------------------------------
List Token :
-----------------------------------------------
TOKEN_PROPOSITION   : 1
TOKEN_NOT           : 2
TOKEN_AND           : 3
TOKEN_OR            : 4
TOKEN_XOR           : 5
TOKEN_IF            : 6
TOKEN_THEN          : 7
TOKEN_IFF           : 8
TOKEN_(             : 9
TOKEN_)             : 10
-----------------------------------------------
Input (String)        : p xor ( q and not ( p and q ) )
Output (Token Lexic)  : 1   5   9   1   3   2   9   1   3   1   10  10
Output                : Valid

Process finished with exit code 0
```

Contoh No.1 tidak valid

```
"C:\Program Files (x86)\Java\jdk1.8.0_102\bin\java" ...
Lexical Analyzer
-----------------------------------------------
Muhammad Faisal Amir (1301198497)
Hildan Fawwaz        (1301198514)
Monica liviandra     (1301198517)
Revi Chandra Riana   (1301198512)
IFX - 43 - 02
-----------------------------------------------
List Token :
-----------------------------------------------
TOKEN_PROPOSITION   : 1
TOKEN_NOT           : 2
TOKEN_AND           : 3
TOKEN_OR            : 4
TOKEN_XOR           : 5
TOKEN_IF            : 6
TOKEN_THEN          : 7
TOKEN_IFF           : 8
TOKEN_(             : 9
TOKEN_)             : 10
-----------------------------------------------
Input (String)        : if p then ( not q s )
Output (Token Lexic)  : 6   1   7   9   2   1   1   10
Output                : Tidak Valid

Process finished with exit code 0
```

Contoh No.2 Tidak Valid

```
"C:\Program Files (x86)\Java\jdk1.8.0_102\bin\java" ...
Lexical Analyzer
-----------------------------------------------
Muhammad Faisal Amir (1301198497)
Hildan Fawwaz        (1301198514)
Monica liviandra     (1301198517)
Revi Chandra Riana   (1301198512)
IFX - 43 - 02
-----------------------------------------------
List Token :
-----------------------------------------------
TOKEN_PROPOSITION   : 1
TOKEN_NOT           : 2
TOKEN_AND           : 3
TOKEN_OR            : 4
TOKEN_XOR           : 5
TOKEN_IF            : 6
TOKEN_THEN          : 7
TOKEN_IFF           : 8
TOKEN_(             : 9
TOKEN_)             : 10
-----------------------------------------------
Input (String)        : ( p and q ifg ( r or s )
Output (Token Lexic)  : 9   1   3   1   error
Output                : Tidak Valid

Process finished with exit code 0
```