LEXICAL ANALYZER DAN PARSER

Disusun Untuk Memenuhi Tugas Besar Mata Kuliah Teori Bahasa Dan Automata

Oleh Kelompok 1:

1. Kurniadi Ahmad Wijaya 1301194024 / IF 43 09
2. Naufal Haritsah Luthfi 1301194073 / IF 43 09
3. Hanvito Michael Lee 1301190090 / IF 43 09

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2021

Daftar Isi

A. Context Free Grammar

Untuk context free grammar pada tugas besar ini digunakan bahasa prancis. Adapun bentuk valid yang mempresentasikan bahasa perancis adalah sebagai berikut.

<S> ::= <SB> <VB> <OB>

<SB> ::= (saya) moi | (ibu) mère | (bapak) père | (dia sebagai laki-laki) il

<VB> ::= (mengendarai) conduire | (menyiram) affleurer | (memakai) porter

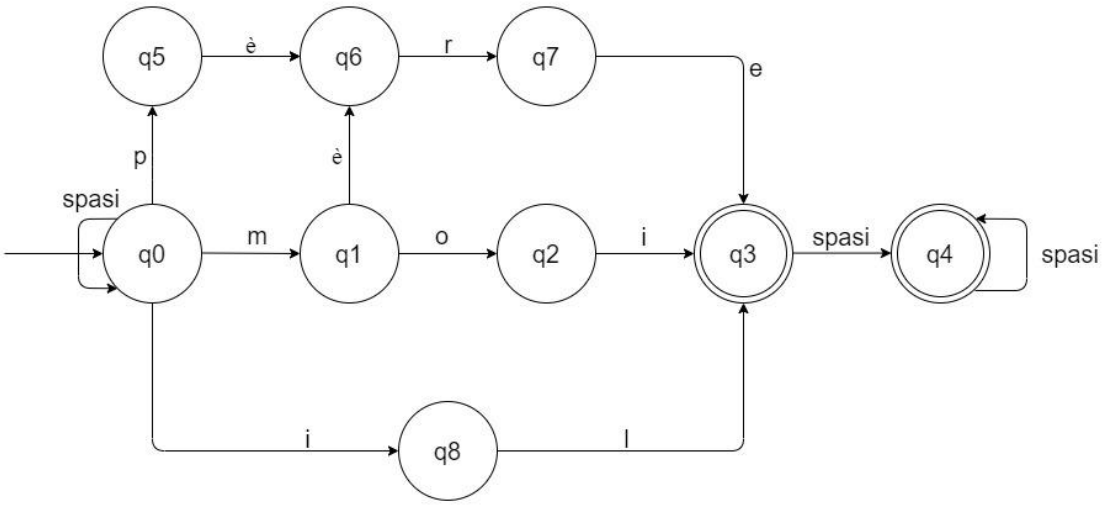
<OB> ::= (mobil) voiture | (tanaman) plante | (cincin) bague

Simbol non-terminal: S (Starting symbol), SB (Subject), VB (Verb), OB (Object)
Simbol terminal: moi, mère, père, il, conduire, affleurer, porter, voiture, plante, bague

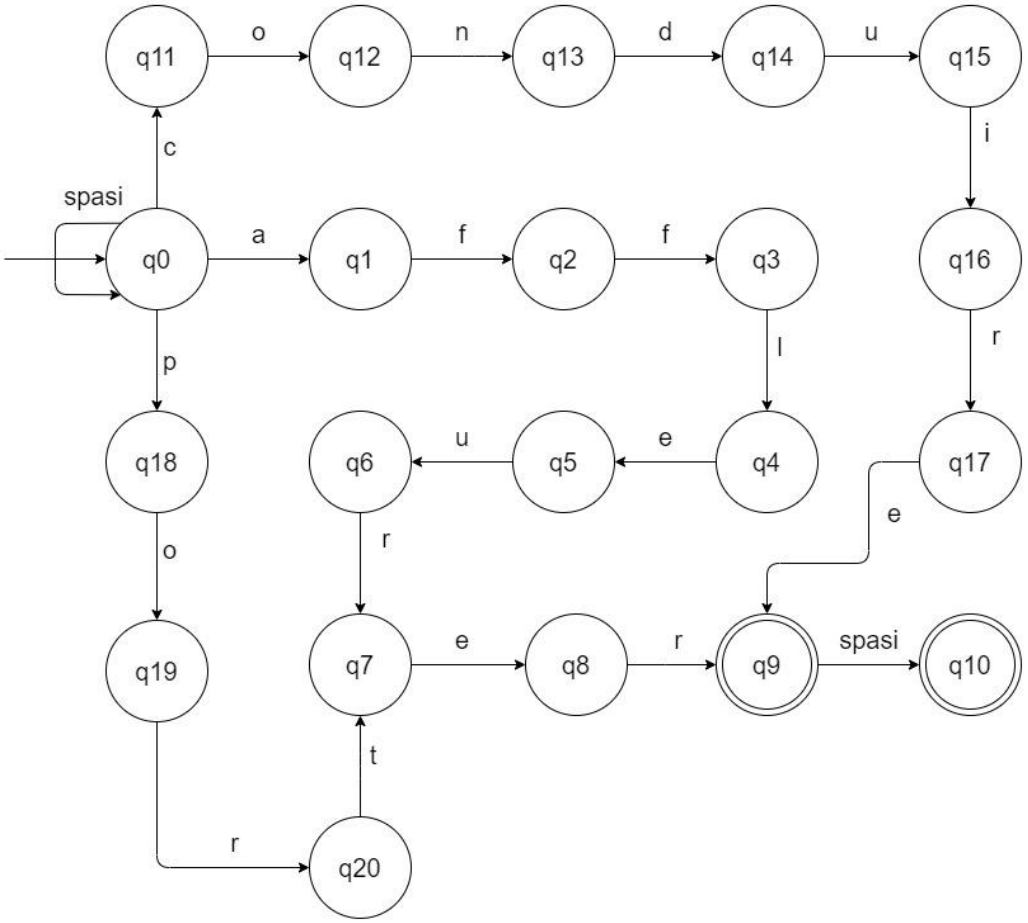
B. Finite Automata

Setelah membuat aturan untuk grammar, langkah selanjutnya yang akan dilakukan adalah membangun finite automata sebagai rules state-state serta value yang akan digunakan pada program lexical analyzer yang akan dibuat. Adapun bentuk model finite automata yang dibuat yaitu:

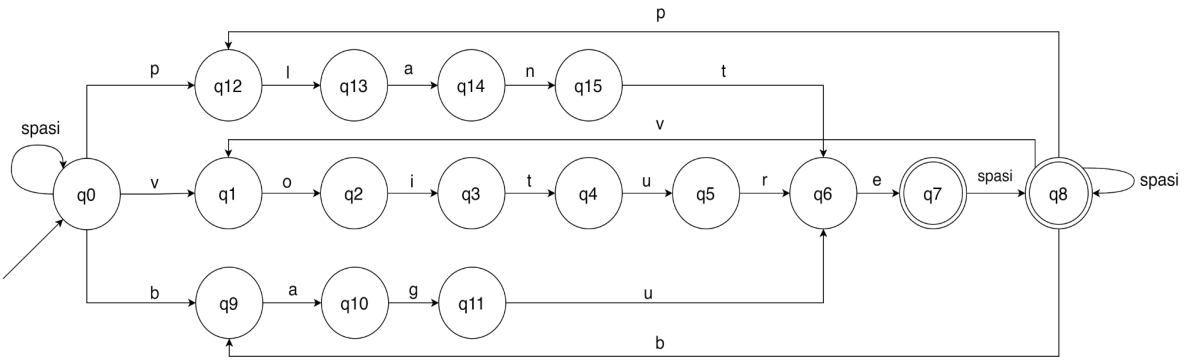
1. Subject



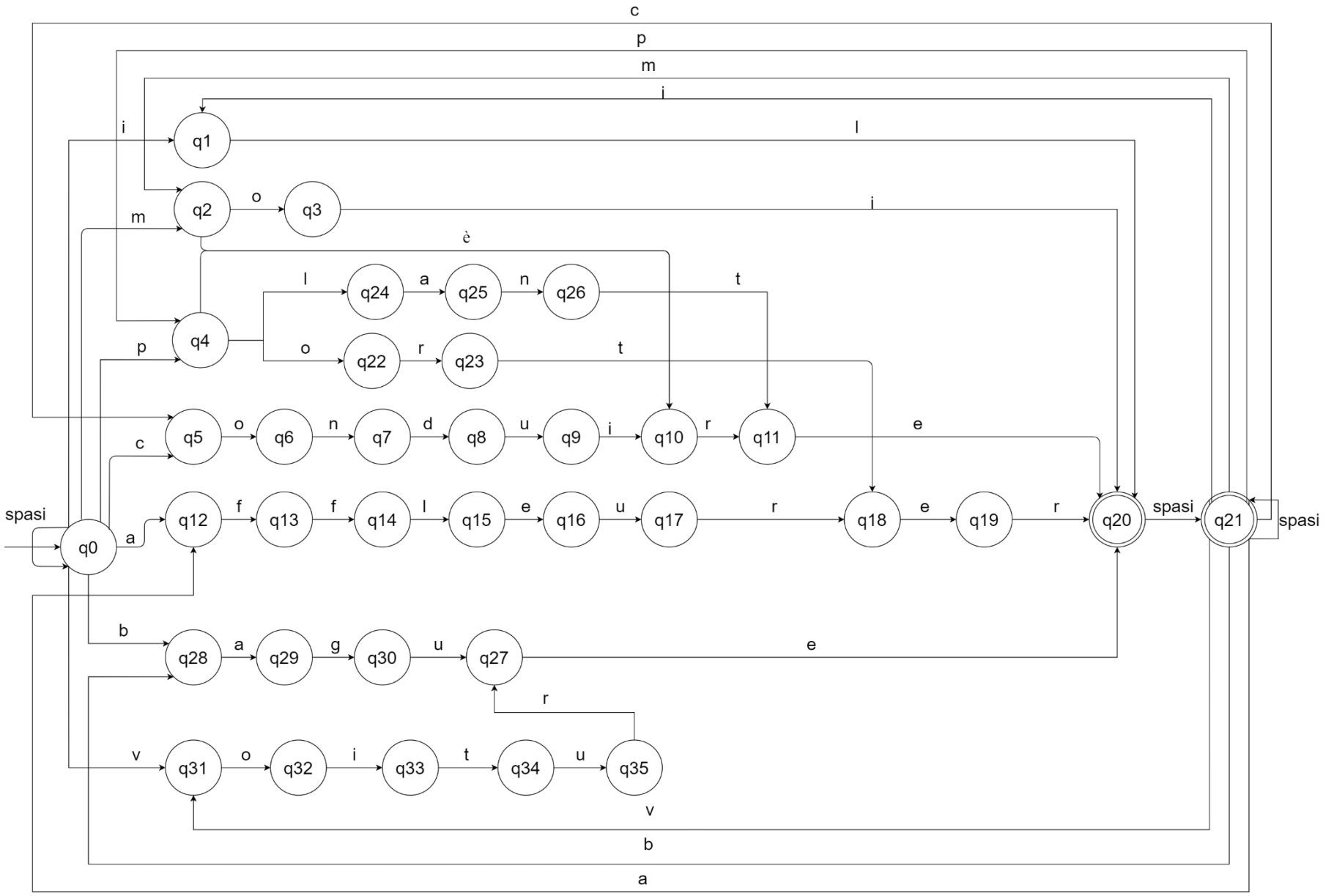
2. Verb



3. Object



4. Hasil Akhir Penggabungan



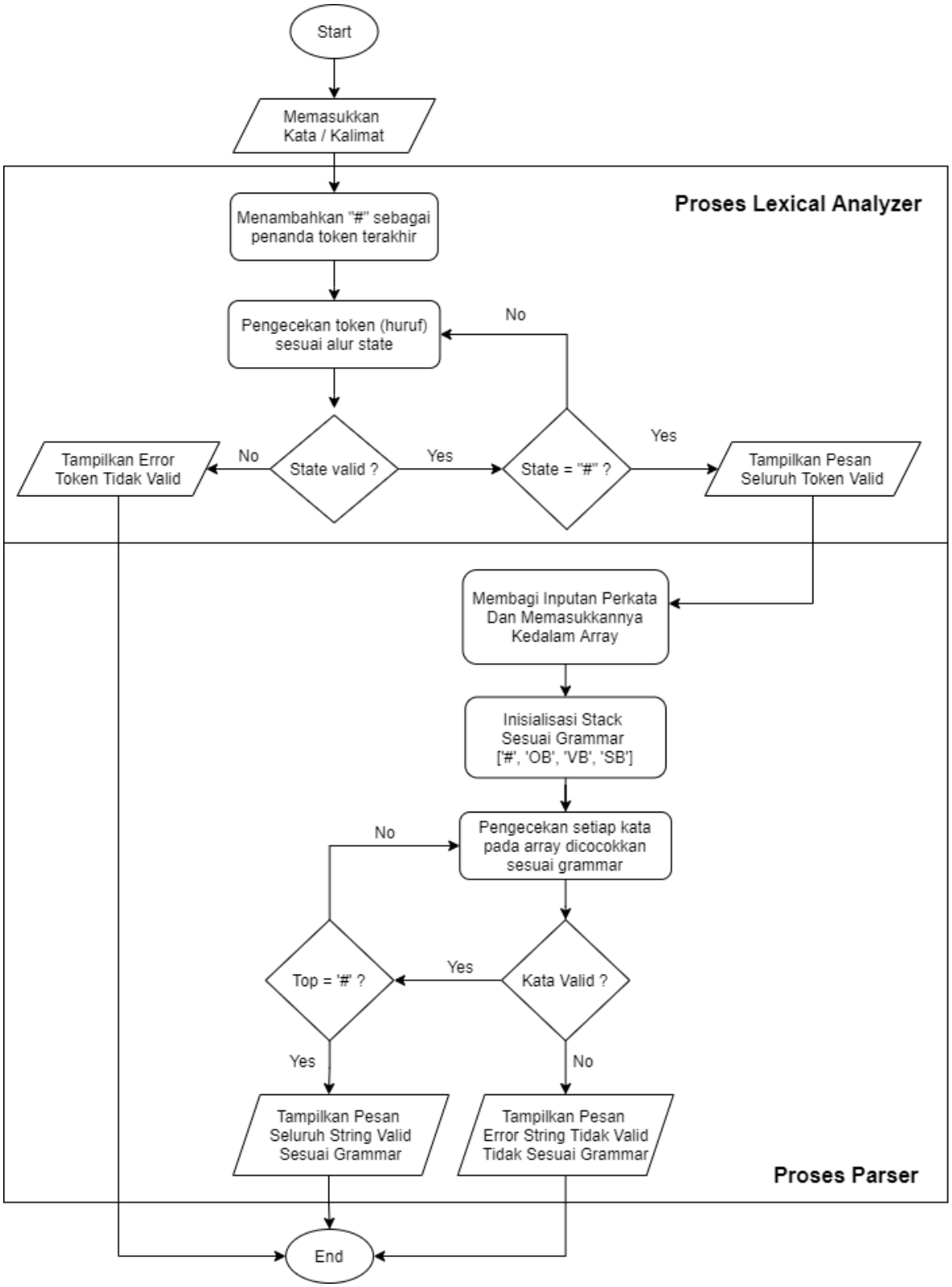
C. Parse Table (LL)

Setelah melakukan penyusunan finite automata yang akan dibangun selanjutnya dibuat parse table yang akan digunakan untuk membangun program parse kalimat untuk menentukan susunan aturan grammar yang telah ditentukan yaitu <subject> <verb> <object> adapun susunan parse table dijabarkan sebagai berikut.

	moi	**mère**	**père**	**il**	**conduire**	**affleurer**	**porter**	**voiture**	**plante**	**bague**	**EOS**
S	SB VB OB	SB VB OB	SB VB OB	SB VB OB	error	error	error	error	error	error	error
SB	moi	mère	père	il	error	error	error	error	error	error	error
VB	error	error	error	error	conduire	affleurer	porter	error	error	error	error
OB	error	error	error	error	error	error	error	voiture	plante	bague	error

D. Alur Program

Adapun alur program Lexical Analyzer Dan Parser yang dibuat hanya sekali input saja sehingga apabila lexical analyzer tidak sukses maka parser tidak akan dijalankan karena error. Adapun secara sederhana proses program dijelaskan melalui flowchart dibawah.



E. Implementasi Program

Program lexical analyzer dan parser ini dibagi menjadi beberapa bagian yaitu:

1. main.py

Program ini merupakan kode utama program yang akan memanggil proses lexical analyzer dan menentukan apakah hasil analisis dapat melakukan running parser atau tidak. Berikut merupakan implementasi kode pada main.py.

```
import lexical
import parse

print("""\
_____________________________________________________________________
                    Lexical Analyzer Dan Parser                     |
_____________________________________________________________________
  Oleh Kelompok 1 IF-43-09: Hanvito Michael Lee   (1301190090)      |
                            Kurniadi Ahmad Wijaya (1301194024)      |
                            Naufal Haritsah Lutfi (1301194073)      |
                                                                    |
  Grammar (Perancis):                                               |
  ___________________________________                               |
  | subject |   verb    |  object   |                               |
  ___________________________________                               |
  |   moi   | conduire  | voiture   |                               |
  |   mère  | affleurer | plante    |                               |
  |   père  | porter    | bague     |                               |
  |   il    |           |           |                               |
  ___________________________________                               |
                                                                    |
  Format Masukkan : <subject> <verb> <object>                       |
  Contoh Masukkan : moi conduire plante                             |
_____________________________________________________________________
""")

kalimat = input("Masukkan Kalimat / Kata Yang Ingin Diperiksa: ")

print("""\
_____________________________________
    Proses Lexical Analyzer
_____________________________________""")
is_valid = lexical.analyze(kalimat)

print("""\
_____________________________________
          Proses Parser
_____________________________________""")
if is_valid:
   parse.analyze(kalimat)
else :
   print(f'Input string {kalimat}, tidak diterima, tidak ada pada Grammar'
```

2. **lexical.py**

Program ini merupakan bagian dimana proses analisis kata untuk ditentukan ada atau tidak pada grammar yang telah dibuat sebelumnya. Pada bagian ini dipanggil 3 buah modul yaitu object.py, subject.py, dan verb.py untuk digunakan fungsi transition_table. Adapun implementasi kode pada lexical.py adalah sebagai berikut.

```
from collections import defaultdict

import grammar.subject as subject
import grammar.object as object
import grammar.verb as verb

def analyze(input_string):
    input_string = input_string.lower() + "#"

    # Inisialisasi State [q0, q1 ...]
    state_list = []; list(state_list.append(f'q{i}') for i in range(32))

    # Inisialisasi Seluruh State Menjadi ERROR
    transition_table = defaultdict(lambda: "ERROR", {})

    # Initial State (q0)
    transition_table[("q0", " ")] = "q0"

    # Finish state
    transition_table[("q20", "#")] = "ACCEPT"
    transition_table[("q20", " ")] = "q21"

    transition_table[("q21", "#")] = "ACCEPT"
    transition_table[("q21", " ")] = "q21"

    # Initialize State
    transition_table = subject.transition_table(transition_table)
    transition_table = object.transition_table(transition_table)
    transition_table = verb.transition_table(transition_table)

    # lexical Analysis
    idx_char = 0
    state = "q0"
    current_token = ""

    while state ≠ "ACCEPT":
        current_char = input_string[idx_char]
        current_token += current_char

        print(state, current_char)
        state = transition_table[(state, current_char)]

        if state == "q20":
            print("current token: {} is valid".format(current_token))
            current_token = ""
        if state == "ERROR":
            print("error")
            break

        idx_char += 1

    # Conclusion
    if state == "ACCEPT":
        print(f'\nsemua token yang di input: {input_string} valid\n')

    return state == "ACCEPT"
```

3. **parse.py**

Program ini merupakan bagian dimana proses analisis kalimat untuk ditentukan apakah sudah sesuai dengan aturan input pada parse table sebelumnya atau tidak. Pada bagian ini dipanggil 3 buah modul yaitu object.py, subject.py, dan verb.py untuk digunakan fungsi parse_table. Adapun implementasi kode pada gr_parser.py adalah sebagai berikut.

```
import grammar.subject as subject
import grammar.object as object
import grammar.verb as verb

def analyze(sentence):
    tokens = sentence.lower().split()
    tokens.append('EOS')

    non_terminals = ['S', 'SB', 'VB', 'OB']
    terminals = [
        'moi', 'mère', 'père', 'il', 'conduire', 'affleurer',
        'porter', 'voiture', 'plante', 'bague'
    ]

    parse_table = {}

    # Non Terminal S
    parse_table[('S', 'moi')] = ['SB', 'VB', 'OB']
    parse_table[('S', 'mère')] = ['SB', 'VB', 'OB']
    parse_table[('S', 'père')] = ['SB', 'VB', 'OB']
    parse_table[('S', 'il')] = ['SB', 'VB', 'OB']
    parse_table[('S', 'conduire')] = ['error']
    parse_table[('S', 'affleurer')] = ['error']
    parse_table[('S', 'porter')] = ['error']
    parse_table[('S', 'voiture')] = ['error']
    parse_table[('S', 'plante')] = ['error']
    parse_table[('S', 'bague')] = ['error']
    parse_table[('S', 'EOS')] = ['error']

    # Non Terminal SB
    parse_table = subject.parse_table(parse_table)

    # Non Terminal VB
    parse_table = verb.parse_table(parse_table)

    # Non Terminal OB
    parse_table = object.parse_table(parse_table)

    stack = []
    stack.append('#')
    stack.append('S')

    token = 0
    symbol = tokens[token]
```

```
while(len(stack) > 0):
    top = stack[len(stack)-1]

    print(f'Top    = {top}')
    print(f'Symbol = {symbol}')

    if top in terminals:
        if top == symbol:
            stack.pop()
            token += 1
            symbol = tokens[token]

            if symbol == 'EOS':
                print('Isi Stack :', stack)
                stack.pop()
        else:
            print('error')
            break
    elif top in non_terminals:
        if parse_table[(top, symbol)][0] != 'error':
            stack.pop()
            push = parse_table[(top, symbol)]

            for i in range(len(push)-1, -1, -1):
                stack.append(push[i])
        else:
            print('error')
            break
    else:
        print('error')
        break

    print(f'Isi Stack : {stack} \n')

if symbol == 'EOS' and len(stack) == 0:
    print(f'Input string {sentence}, diterima sesuai Grammar')
else:
    print(f'Input string {sentence}, tidak diterima, tidak sesuai Grammar')
```

4. **object.py**

   Program ini merupakan bagian yang memetakkan tabel transisi dan tabel parser untuk setiap kata pada object sesuai dengan Finite Automata yang telah dibuat.

```
def transition_table(transition):
    # string "voiture"
    transition[("q21", "v")] = "q31"
    transition[("q0", "v")] = "q31"
    transition[("q31", "o")] = "q32"
    transition[("q32", "i")] = "q33"
    transition[("q33", "t")] = "q34"
    transition[("q34", "u")] = "q35"
    transition[("q35", "r")] = "q27"
    transition[("q27", "e")] = "q20"

    # string "plante"
    transition[("q21", "p")] = "q4"
    transition[("q0", "p")] = "q4"
    transition[("q4", "l")] = "q24"
    transition[("q24", "a")] = "q25"
    transition[("q25", "n")] = "q26"
    transition[("q26", "t")] = "q11"
    transition[("q11", "e")] = "q20"

    # string "bague"
    transition[("q21", "b")] = "q28"
    transition[("q0", "b")] = "q28"
    transition[("q28", "a")] = "q29"
    transition[("q29", "g")] = "q30"
    transition[("q30", "u")] = "q27"
    transition[("q27", "e")] = "q20"

    return transition

def parse_table(parse):
    parse[('OB', 'moi')] = ['error']
    parse[('OB', 'mère')] = ['error']
    parse[('OB', 'père')] = ['error']
    parse[('OB', 'il')] = ['error']
    parse[('OB', 'conduire')] = ['error']
    parse[('OB', 'affleurer')] = ['error']
    parse[('OB', 'porter')] = ['porter']
    parse[('OB', 'voiture')] = ['voiture']
    parse[('OB', 'plante')] = ['plante']
    parse[('OB', 'bague')] = ['bague']
    parse[('OB', 'EOS')] = ['error']

    return parse
```

5. **subject.py**

   Program ini merupakan bagian yang memetakkan tabel transisi dan tabel parser untuk setiap kata pada subject sesuai dengan Finite Automata yang telah dibuat.

```
def transition_table(transition):
    # string "moi"
    transition[("q21", "m")] = "q2"
    transition[("q0", "m")] = "q2"
    transition[("q2", "o")] = "q3"
    transition[("q3", "i")] = "q20"

    # string "mère"
    transition[("q21", "m")] = "q2"
    transition[("q0", "m")] = "q2"
    transition[("q2", "è")] = "q10"
    transition[("q10", "r")] = "q11"
    transition[("q11", "e")] = "q20"

    # string "père"
    transition[("q21", "p")] = "q4"
    transition[("q0", "p")] = "q4"
    transition[("q4", "è")] = "q10"
    transition[("q10", "r")] = "q11"
    transition[("q11", "e")] = "q20"

    # string "il"
    transition[("q21", "i")] = "q1"
    transition[("q0", "i")] = "q1"
    transition[("q1", "l")] = "q20"

    return transition


def parse_table(parse):
    parse[('SB', 'moi')] = ['moi']
    parse[('SB', 'mère')] = ['mère']
    parse[('SB', 'père')] = ['père']
    parse[('SB', 'il')] = ['il']
    parse[('SB', 'conduire')] = ['error']
    parse[('SB', 'affleurer')] = ['error']
    parse[('SB', 'porter')] = ['error']
    parse[('SB', 'voiture')] = ['error']
    parse[('SB', 'plante')] = ['error']
    parse[('SB', 'bague')] = ['error']
    parse[('SB', 'EOS')] = ['error']

    return parse
```

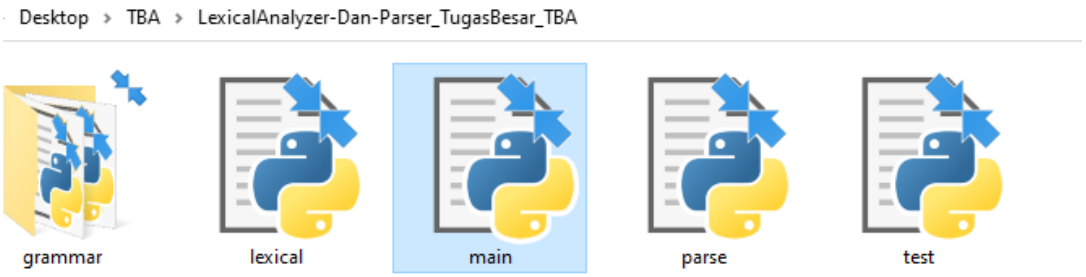
6. **verb.py**

Program ini merupakan bagian yang memetakkan tabel transisi dan tabel parser untuk setiap kata pada verb sesuai dengan Finite Automata yang telah dibuat..

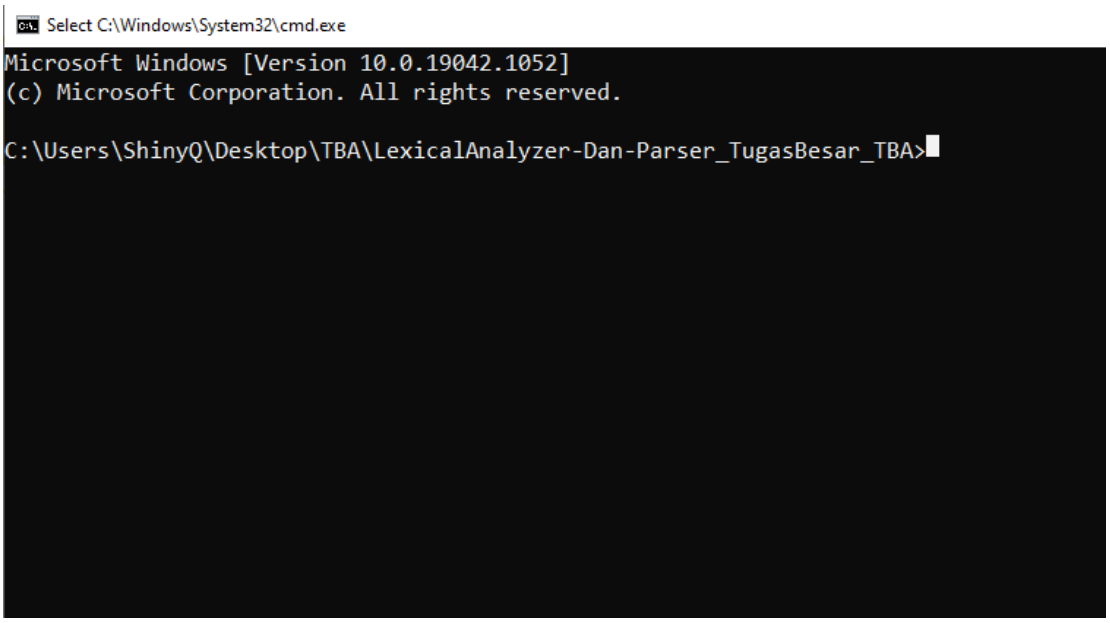
```
def transition_table(transition):
    # string "conduire"
    transition[("q21", "c")] = "q5"
    transition[("q0", "c")] = "q5"
    transition[("q5", "o")] = "q6"
    transition[("q6", "n")] = "q7"
    transition[("q7", "d")] = "q8"
    transition[("q8", "u")] = "q9"
    transition[("q9", "i")] = "q10"
    transition[("q10", "r")] = "q11"
    transition[("q11", "e")] = "q20"

    # string "affleurer"
    transition[("q21", "a")] = "q12"
    transition[("q0", "a")] = "q12"
    transition[("q12", "f")] = "q13"
    transition[("q13", "f")] = "q14"
    transition[("q14", "l")] = "q15"
    transition[("q15", "e")] = "q16"
    transition[("q16", "u")] = "q17"
    transition[("q17", "r")] = "q18"
    transition[("q18", "e")] = "q19"
    transition[("q19", "r")] = "q20"

    # string "porter"
    transition[("q21", "p")] = "q4"
    transition[("q0", "p")] = "q4"
    transition[("q4", "o")] = "q22"
    transition[("q22", "r")] = "q23"
    transition[("q23", "t")] = "q18"
    transition[("q18", "e")] = "q19"
    transition[("q19", "r")] = "q20"

    return transition


def parse_table(parse):
    parse[('VB', 'moi')] = ['error']
    parse[('VB', 'mère')] = ['error']
    parse[('VB', 'père')] = ['error']
    parse[('VB', 'il')] = ['error']
    parse[('VB', 'conduire')] = ['conduire']
    parse[('VB', 'affleurer')] = ['affleurer']
    parse[('VB', 'porter')] = ['porter']
    parse[('VB', 'voiture')] = ['error']
    parse[('VB', 'plante')] = ['error']
    parse[('VB', 'bague')] = ['error']
    parse[('VB', 'EOS')] = ['error']

    return parse
```

F. Langkah Penggunaan Program

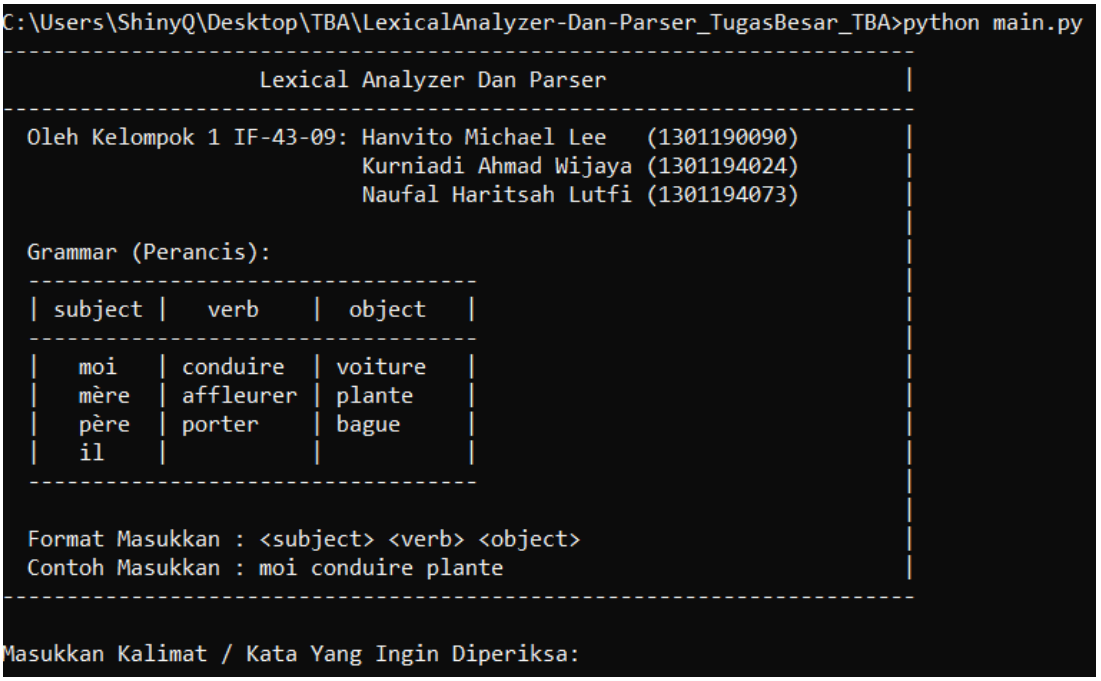
1. Buka Direktori File Berada



2. Jalankan terminal Command Prompt pada direktori folder tersebut



3. Jalankan program dengan memasukkan command python main.py



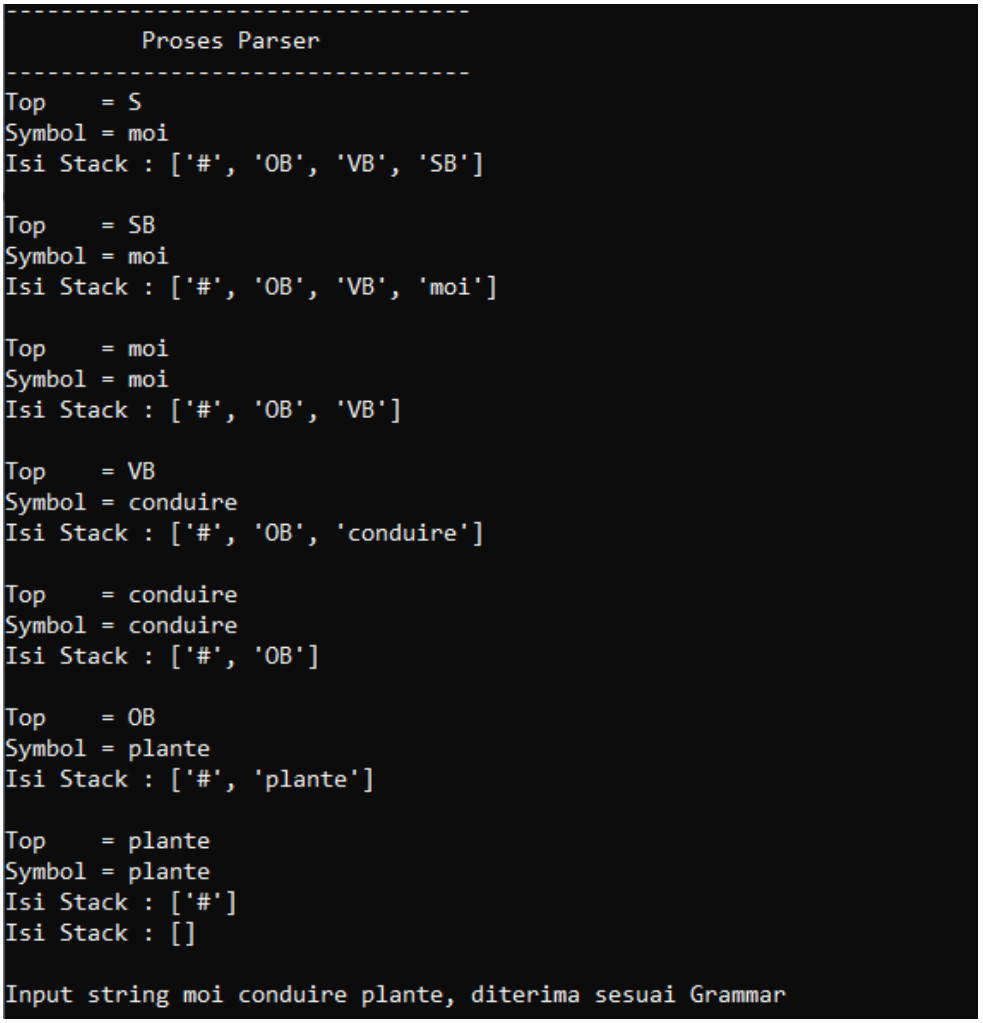
4. Setelah itu kita dapat langsung memasukkan kata / kalimat dengan melihat referensi pada tabel yang tertera di instruksi. Setelah memasukkan dan menekan enter maka proses lexical analyzer akan berjalan. Jika kata tidak valid atau tidak ada pada grammar maka program akan terhenti.

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: moi conduire plante
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 m
q2 o
q3 i
current token: moi is valid
q20
q21 c
q5 o
q6 n
q7 d
q8 u
q9 i
q10 r
q11 e
current token:  conduire is valid
q20
q21 p
q4 l
q24 a
q25 n
q26 t
q11 e
current token:  plante is valid
q20 #

semua token yang di input: moi conduire plante# valid
```

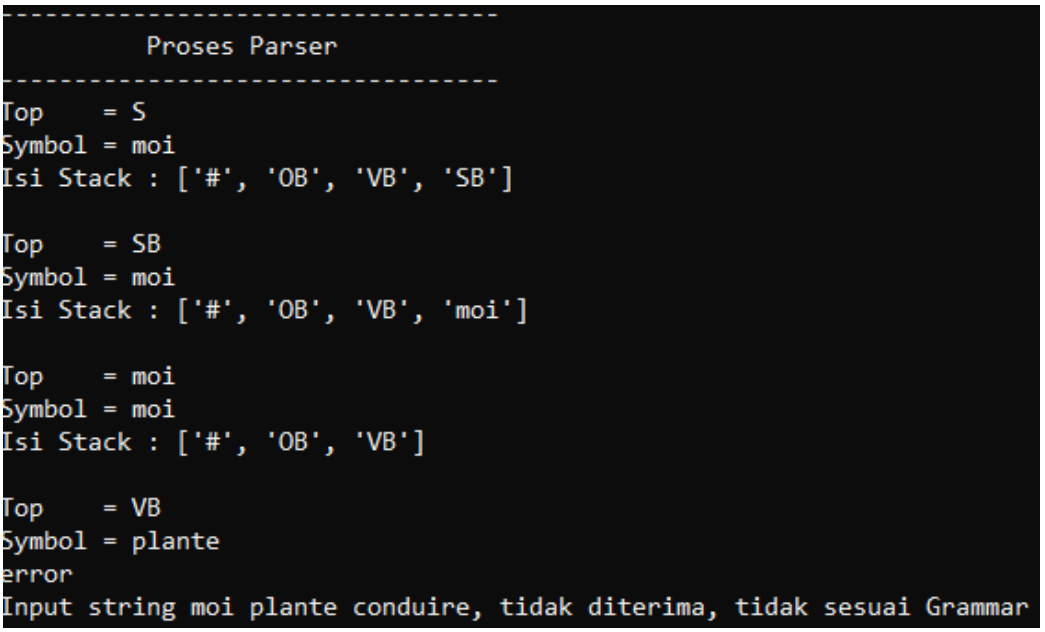
Contoh hasil salah:

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: moi condd sdsd
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 m
q2 o
q3 i
current token: moi is valid
q20
q21 c
q5 o
q6 n
q7 d
q8 d
error
---------------------------------
          Proses Parser
---------------------------------
Input string moi condd sdsd, tidak diterima, tidak ada pada Grammar
```

5. Apabila proses lexical analyzer berhasil maka program akan langsung melanjutkan ke proses parsing. Adapun hasil dari proses parse adalah sebagai berikut



Jika terdapat kesalahan parse yaitu susunan grammar yang salah maka akan muncul error.



G. Hasil Keluaran

1. Lexical Analyzer

1.1 Contoh Valid Input

Input = moi

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: moi
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 m
q2 o
q3 i
current token: moi is valid
q20 #
```

Inputan “moi” bernilai valid karena string “moi” terdapat pada grammar yang sudah didefinisikan.

Input = “conduire”

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: conduire
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 c
q5 o
q6 n
q7 d
q8 u
q9 i
q10 r
q11 e
current token: conduire is valid
q20 #
```

Inputan “voiture” bernilai valid karena string “moi” terdapat pada grammar yang sudah didefinisikan.

Input = “voiture”

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: voiture
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 v
q31 o
q32 i
q33 t
q34 u
q35 r
q27 e
current token: voiture is valid
q20 #
```

Inputan “voiture” bernilai valid karena string “moi” terdapat pada grammar yang sudah didefinisikan.

2.1 Contoh Invalid Input

Input = “mere”

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: mere
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 m
q2 e
error
```

Inputan “mere” bernilai invalid karena pada grammar terdapat string “mère”, bukan string “mere”.

Input = “grande”

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: grande
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 g
error
```

Inputan “grande” bernilai invalid karena string tersebut tidak terdapat pada grammar.

Input =”voiturer”

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: voiturer
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 v
q31 o
q32 i
q33 t
q34 u
q35 r
q27 e
current token: voiture is valid
q20 r
error
```

Inputan “voiturer” bernilai invalid karena string tersebut tidak terdapat pada grammar.

2. Parser

1.1 Contoh Valid Input

Input = “il conduire voiture”

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: il conduire voiture
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 i
q1 l
current token: il is valid
q20
q21 c
q5 o
q6 n
q7 d
q8 u
q9 i
q10 r
q11 e
current token:  conduire is valid
q20
q21 v
q31 o
q32 i
q33 t
q34 u
q35 r
q27 e
current token:  voiture is valid
q20 #

semua token yang di input: il conduire voiture# valid
```

```
---------------------------------
          Proses Parser
---------------------------------
Top    = S
Symbol = il
Isi Stack : ['#', 'OB', 'VB', 'SB']

Top    = SB
Symbol = il
Isi Stack : ['#', 'OB', 'VB', 'il']

Top    = il
Symbol = il
Isi Stack : ['#', 'OB', 'VB']

Top    = VB
Symbol = conduire
Isi Stack : ['#', 'OB', 'conduire']

Top    = conduire
Symbol = conduire
Isi Stack : ['#', 'OB']

Top    = OB
Symbol = voiture
Isi Stack : ['#', 'voiture']

Top    = voiture
Symbol = voiture
Isi Stack : ['#']
Isi Stack : []

Input string il conduire voiture, diterima sesuai Grammar
```

Inputan “il conduire voiture” bernilai valid karena string tersebut sudah memenuhi CFG yang telah dibuat.

Input = “mère porter bague”

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: mère porter bague
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 m
q2 è
q10 r
q11 e
current token: mère is valid
q20
q21 p
q4 o
q22 r
q23 t
q18 e
q19 r
current token:  porter is valid
q20
q21 b
q28 a
q29 g
q30 u
q27 e
current token:  bague is valid
q20 #

semua token yang di input: mère porter bague# valid
```

```
---------------------------------
          Proses Parser
---------------------------------
Top    = S
Symbol = mère
Isi Stack : ['#', 'OB', 'VB', 'SB']

Top    = SB
Symbol = mère
Isi Stack : ['#', 'OB', 'VB', 'mère']

Top    = mère
Symbol = mère
Isi Stack : ['#', 'OB', 'VB']

Top    = VB
Symbol = porter
Isi Stack : ['#', 'OB', 'porter']

Top    = porter
Symbol = porter
Isi Stack : ['#', 'OB']

Top    = OB
Symbol = bague
Isi Stack : ['#', 'bague']

Top    = bague
Symbol = bague
Isi Stack : ['#']
Isi Stack : []

Input string mère porter bague, diterima sesuai Grammar
```

Inputan “mère porter bague” bernilai valid karena string tersebut sudah memenuhi CFG yang telah dibuat.

Input = “moi affleurer plante”

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: moi affleurer plante
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 m
q2 o
q3 i
current token: moi is valid
q20
q21 a
q12 f
q13 f
q14 l
q15 e
q16 u
q17 r
q18 e
q19 r
current token:  affleurer is valid
q20
q21 p
q4 l
q24 a
q25 n
q26 t
q11 e
current token:  plante is valid
q20 #

semua token yang di input: moi affleurer plante# valid
```

```
---------------------------------
          Proses Parser
---------------------------------
Top    = S
Symbol = moi
Isi Stack : ['#', 'OB', 'VB', 'SB']

Top    = SB
Symbol = moi
Isi Stack : ['#', 'OB', 'VB', 'moi']

Top    = moi
Symbol = moi
Isi Stack : ['#', 'OB', 'VB']

Top    = VB
Symbol = affleurer
Isi Stack : ['#', 'OB', 'affleurer']

Top    = affleurer
Symbol = affleurer
Isi Stack : ['#', 'OB']

Top    = OB
Symbol = plante
Isi Stack : ['#', 'plante']

Top    = plante
Symbol = plante
Isi Stack : ['#']
Isi Stack : []

Input string moi affleurer plante, diterima sesuai Grammar
```

Inputan “mère porter bague” bernilai valid karena string tersebut sudah memenuhi CFG yang telah dibuat.

2.1 Contoh Invalid Input

Input = “moi moi moi”

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: moi moi moi
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 m
q2 o
q3 i
current token: moi is valid
q20
q21 m
q2 o
q3 i
current token:  moi is valid
q20
q21 m
q2 o
q3 i
current token:  moi is valid
q20 #

semua token yang di input: moi moi moi# valid
```

```
---------------------------------
          Proses Parser
---------------------------------
Top    = S
Symbol = moi
Isi Stack : ['#', 'OB', 'VB', 'SB']

Top    = SB
Symbol = moi
Isi Stack : ['#', 'OB', 'VB', 'moi']

Top    = moi
Symbol = moi
Isi Stack : ['#', 'OB', 'VB']

Top    = VB
Symbol = moi
error
Input string moi moi moi, tidak diterima, tidak sesuai Grammar
```

Inputan “moi moi moi” bernilai invalid karena string tersebut tidak memenuhi parse table dan CFG yang telah dibuat.

Input = “conduire il bague”

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: conduire il bague
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 c
q5 o
q6 n
q7 d
q8 u
q9 i
q10 r
q11 e
current token: conduire is valid
q20
q21 i
q1 l
current token:  il is valid
q20
q21 b
q28 a
q29 g
q30 u
q27 e
current token:  bague is valid
q20 #

semua token yang di input: conduire il bague# valid
```

```
---------------------------------
          Proses Parser
---------------------------------
Top    = S
Symbol = conduire
error
Input string conduire il bague, tidak diterima, tidak sesuai Grammar
```

Inputan “conduire il bague” bernilai invalid karena string tersebut tidak memenuhi parse table dan CFG yang telah dibuat.

Input = “mère affleurer moi”

```
Masukkan Kalimat / Kata Yang Ingin Diperiksa: mère affleurer moi
---------------------------------
    Proses Lexical Analyzer
---------------------------------
q0 m
q2 è
q10 r
q11 e
current token: mère is valid
q20
q21 a
q12 f
q13 f
q14 l
q15 e
q16 u
q17 r
q18 e
q19 r
current token:  affleurer is valid
q20
q21 m
q2 o
q3 i
current token:  moi is valid
q20 #

semua token yang di input: mère affleurer moi# valid
```

```
---------------------------------
         Proses Parser
---------------------------------
Top    = S
Symbol = mère
Isi Stack : ['#', 'OB', 'VB', 'SB']

Top    = SB
Symbol = mère
Isi Stack : ['#', 'OB', 'VB', 'mère']

Top    = mère
Symbol = mère
Isi Stack : ['#', 'OB', 'VB']

Top    = VB
Symbol = affleurer
Isi Stack : ['#', 'OB', 'affleurer']

Top    = affleurer
Symbol = affleurer
Isi Stack : ['#', 'OB']

Top    = OB
Symbol = moi
error
Input string mère affleurer moi, tidak diterima, tidak sesuai Grammar
```

Inputan “mère affleurer moi” bernilai invalid karena string tersebut tidak memenuhi parse table dan CFG yang telah dibuat.