Lexical Analysis dan Parser Sederhana untuk Teks Bahasa Alami

dibuat untuk memenuhi tugas besar mata kuliah Teori Bahasa dan Automata

Oleh:

Bagus Seno Pamungkas	(1301190337)
Fauzi Arya Surya Abadi	(1301194101)
Anindika Riska Intan Fauzy	(1301194254)

Semester Genap 2020/2021

Program Studi S1 Informatika
Fakultas Informatika
Universitas Telkom
Bandung

1. Pendahuluan

Seperti halnya pada tata bahasa regular, sebuah Tata Bahasa Bebas Konteks (*Context Free Grammar* atau CFG) adalah suatu cara yang menunjukkan bagaimana menghasilkan untai-untai dalam sebuah bahasa. Seperti kita ketahui, pada saat menurunkan suatu string, simbol-simbol variabel akan mewakili bagian-bagian yang belum yang belum diturunkan dari string tersebut. Bila pada tata bahasa regular, bagian yang belum terturunkan tersebut selalu terjadi pada suatu ujung, pada *context free grammar* bisa terdapat lebih banyak bagian yang belum terturunkan itu dan bisa terjadi dimana saja. Ketika penurunan itu sudah lengkap, semua bagian yang belum terturunkan telah diganti oleh string-string (yang mungkin saja kosong) dari himpunan simbol terminal. Bahasa bebas konteks menjadi dasar dalam pembentukan suatu *parser*/proses analisis sintaksis. Bagian sintaks dalam suatu kompilator kebanyakan didefinisikan dalam tata bahasa bebas konteks. *Context free grammar* sederhana pada laporan ini dibuat dengan representasi aturan atau sintaks kalimat dalam sebuah bahasa bahasa Inggris.

2. Kajian Pustaka

2.1 Context Free Grammar

Context Free Grammar atau dapat disingkat menjadi CFG adalah tata bahasa yang mempunyai tujuan sama seperti halnya tata bahasa regular yaitu merupakan suatu cara untuk menunjukkan bagaimana menghasilkan suatu untai-untai dalam sebuah bahasa. CFG dapat disebut juga Tata Bahasa Bebas Konteks yang dapat diartikan sebagai sebuah tata bahasa dimana tidak terdapat pembatasan pada hasil produksinya. Suatu tata bahasa bebas konteks dapat berbentuk sangat melebar, sangat menyempit, atau terjadi rekursif kiri, yang semuanya sering dinamakan bentuk tidak formal. *Context free grammar* digunakan dalam spesifikasi bahasa komputer (pemrograman, script, printer, markup, kamus data, query, perintah).

2.2 Lexical Analysis

Lexical Analysis adalah sebuah proses yang mendahului parsing sebuah rangkaian karakter yang dilakukan di tahapan pertama pada compiler. Ia menerima masukan serangkaian karakter dan menghasilkan deretan simbol yang masing-masing dinamakan token, proses parsing akan lebih mudah dilakukan bila inputnya sudah berupa token. *Lexical analysis* dapat disebut sebagai scanner yang berarti dapat mengubah deretan karakter-karakter menjadi deretan token-token. Proses yang dilakukan pada tahapan ini adalah membaca program sumber karakter per karakter. Satu atau lebih (deretan) karakter-karakter ini dikelompokkan menjadi suatu kesatuan mengikuti pola kesatuan kelompok karakter (token) yang ditentukan dalam bahasa sumber dan disimpan dalam tabel simbol, sedangkan karakter yang tidak mengikuti pola akan dilaporkan sebagai token tak dikenal atau tidak valid.

2.3 Parser

Parser adalah komponen kompiler atau juru bahasa memecah data menjadi elemen yang lebih kecil untuk memudahkan terjemahan ke bahasa lain. Penguraian atau parsing adalah suatu cara memecah-mecah suatu rangkaian masukan dengan mengambil input dalam bentuk urutan token atau intruksi program dan menghasilkan struktur data dalam bentuk pohon uraian (*parse*) atau pohon sintaksis abstrak yang akan digunakan pada tahap kompilasi berikutnya yaitu analisis semantik.

3. Analisis dan Perancangan

3.1 Context Free Grammar

Deskripsi CFG untuk Bahasa Inggris

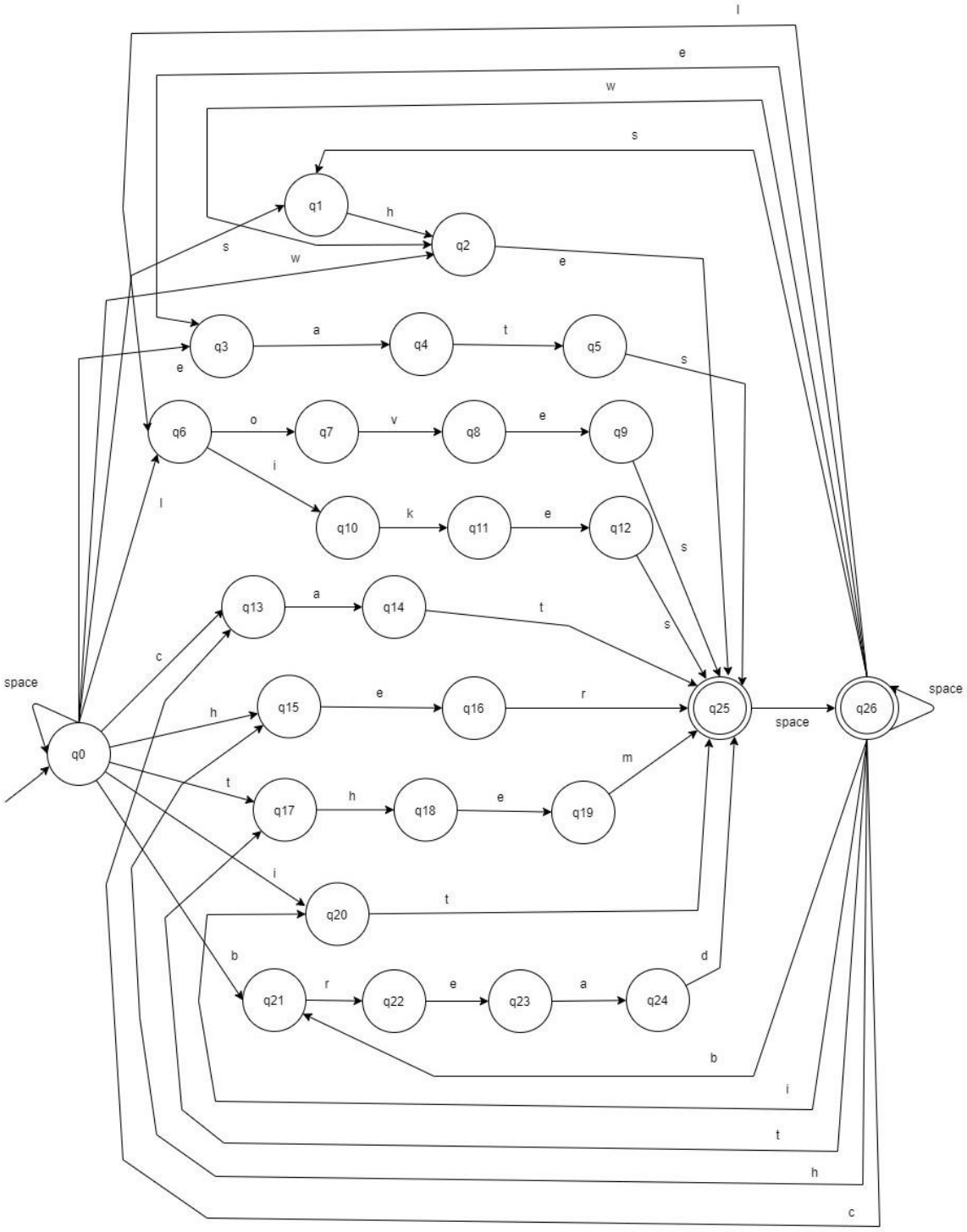
S = {SB, VB, OB}

SB → we | she

VB → eats | loves | likes

OB → her | them | cat | it | bread

3.2 Finite Automata



3.3 Parse Table LL (1)

	she	we	eats	loves	likes	her	them	it	cat	bread	EOS
S	SB VB OB	SB VB OB	error	error	error	SB VB OB	SB VB OB	SB VB OB	SB VB OB	SB VB OB	error
SB	she	he	error	error	error	error	error	error	error	error	error
VB	error	error	eats	loves	likes	error	error	error	error	error	error
OB	error	error	error	error	error	her	them	it	cat	bread	error

4. Hasil

4.1 Lexical Analysis

Berikut adalah kodingan untuk *lexical analysis* untuk menguji kata masukan yang telah ditentukan. Adapun kata yang akan diuji validitasnya dalam *lexical analysis* yaitu *we*, *she*, *eats*, *loves*, *likes*, *her*, *them*, *cat*, *it*, dan *bread*. Untuk kata *we* dan *she* merupakan kata untuk *subject*. Untuk kata *eats*, *loves*, dan *likes* merupakan kata untuk kata kerja (*verb*). Untuk kata *her*, *them*, *cat*, *it*, dan *bread* merupakan kata untuk *object*.

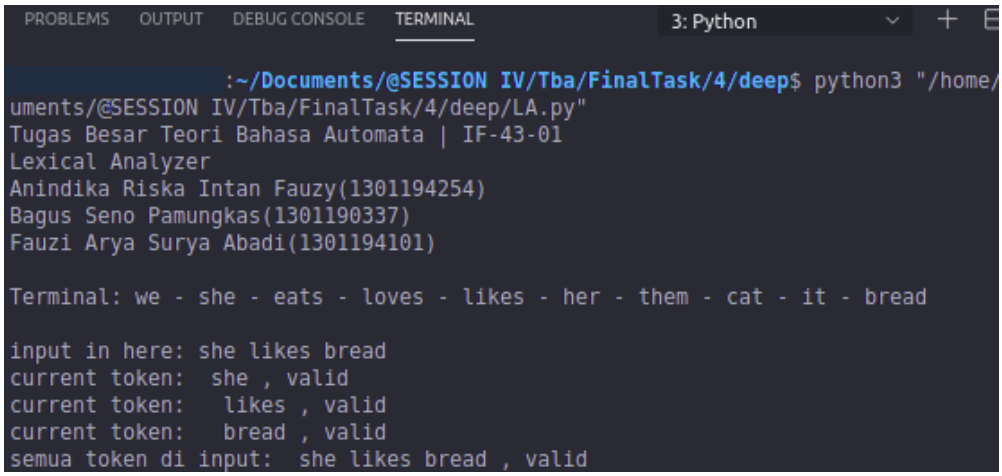
```
print("Tugas Besar Teori Bahasa Automata | IF-43-01")
print("Lexical Analyzer & Parser")
print("Anindika Riska Intan Fauzy (1301194254)\nBagus Seno Pamungkas (1301190337)\nFauzi Arya Surya Abadi (1301194101)\n")

import string

def LA(sentence):
    # initialization
    print("\n====== Lexical Analyzer ====== \n")
    alphabet_list = list(string.ascii_lowercase)
    state_list = [
                'q0','q1','q2','q3','q4','q5','q6','q7','q8','q9','q10',
                'q11','q12','q13','q14','q15','q16','q17','q18','q19',
                'q20','q21','q22','q23','q24','q25','q26',
                ]
    transition_table = {}

    for state in state_list:
        for alphabet in alphabet_list:
            transition_table[(state, alphabet)] = 'error'
        transition_table[(state, '#')] = 'error'
        transition_table[(state, ' ')] = 'error'

    # Deskripsi CFG untuk Bahasa Inggris
    # S = {SB, VB, OB}
    # SB → we | she
    # VB →  eats | loves | likes
    # OB →  her | them | cat | it | bread

    # first state
    transition_table[("q0", " ")] = "q0"

    # Finish state
    transition_table[("q25", "#")] = "accept"
    transition_table[("q25", " ")] = "q26"

    transition_table[("q26", "#")] = "accept"
    transition_table[("q26", " ")] = "q26"
```

```
    #____for string "we"
    transition_table[("q26", "w")] = "q2"
    transition_table[("q0", "w")] = "q2"
    transition_table[("q2", "e")] = "q25"
    transition_table[("q25", " ")] = "q26"

    #____for string "she"
    transition_table[("q26", "s")] = "q1"
    transition_table[("q0", "s")] = "q1"
    transition_table[("q1", "h")] = "q2"
    transition_table[("q2", "e")] = "q25"
    transition_table[("q25", " ")] = "q26"

    #____for string "eats"
    transition_table[("q26", "e")] = "q3"
    transition_table[("q0", "e")] = "q3"
    transition_table[("q3", "a")] = "q4"
    transition_table[("q4", "t")] = "q5"
    transition_table[("q5", "s")] = "q25"
    transition_table[("q25", " ")] = "q26"

    #____for string "loves"
    transition_table[("q26", "l")] = "q6"
    transition_table[("q0", "l")] = "q6"
    transition_table[("q6", "o")] = "q7"
    transition_table[("q7", "v")] = "q8"
    transition_table[("q8", "e")] = "q9"
    transition_table[("q9", "s")] = "q25"
    transition_table[("q25", " ")] = "q26"

    #____for string "likes"
    transition_table[("q26", "l")] = "q6"
    transition_table[("q0", "l")] = "q6"
    transition_table[("q6", "i")] = "q10"
    transition_table[("q10", "k")] = "q11"
    transition_table[("q11", "e")] = "q12"
    transition_table[("q12", "s")] = "q25"
    transition_table[("q25", " ")] = "q26"
```

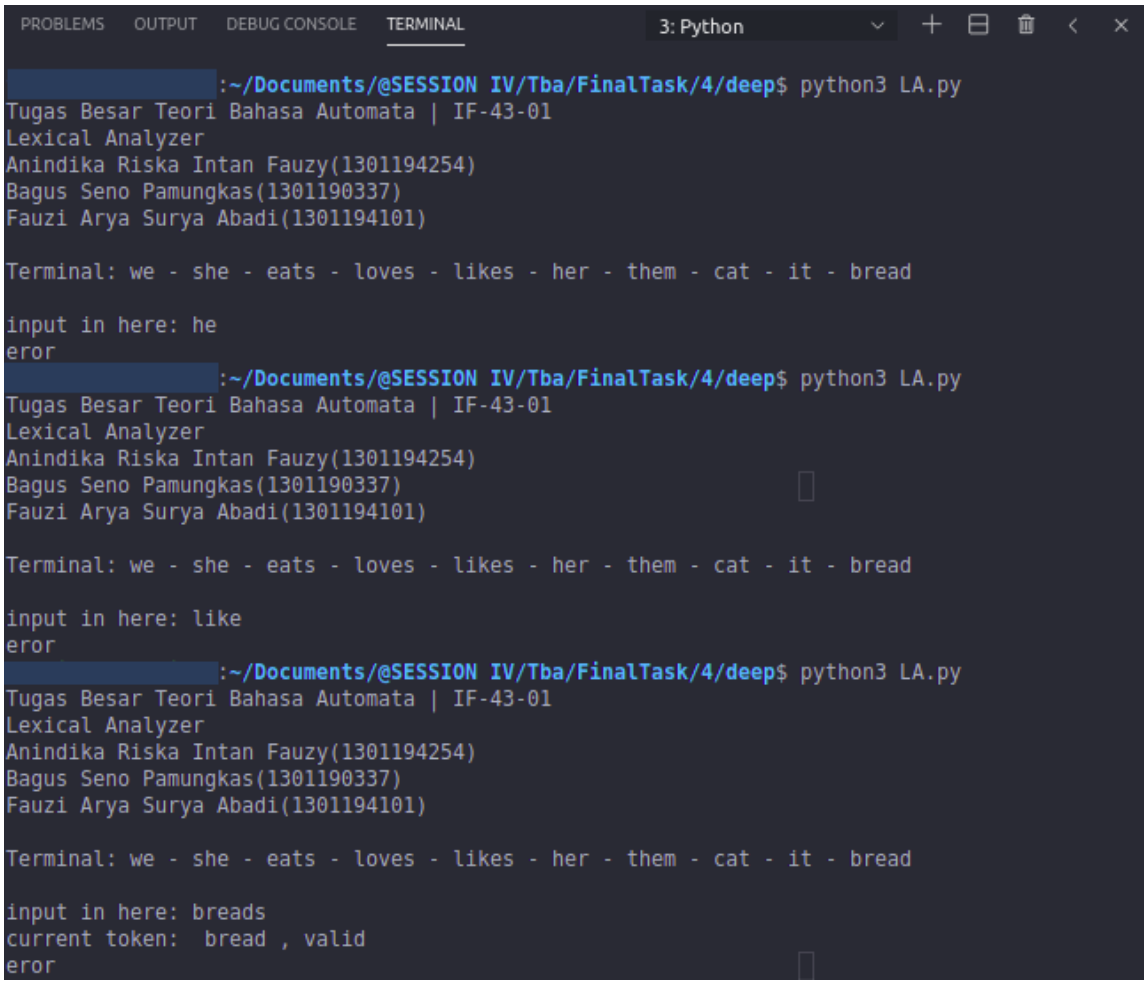
```
    #____for string "cat"
    transition_table[("q26", "c")] = "q13"
    transition_table[("q0", "c")] = "q13"
    transition_table[("q13", "a")] = "q14"
    transition_table[("q14", "t")] = "q25"
    transition_table[("q25", " ")] = "q26"

    #____for string "her"
    transition_table[("q26", "h")] = "q15"
    transition_table[("q0", "h")] = "q15"
    transition_table[("q15", "e")] = "q16"
    transition_table[("q16", "r")] = "q25"
    transition_table[("q25", " ")] = "q26"

    #____for string "them"
    transition_table[("q26", "t")] = "q17"
    transition_table[("q0", "t")] = "q17"
    transition_table[("q17", "h")] = "q18"
    transition_table[("q18", "e")] = "q19"
    transition_table[("q19", "m")] = "q25"
    transition_table[("q25", " ")] = "q26"

    #____for string "it"
    transition_table[("q26", "i")] = "q20"
    transition_table[("q0", "i")] = "q20"
    transition_table[("q20", "t")] = "q25"
    transition_table[("q25", " ")] = "q26"

    #____for string "bread"
    transition_table[("q26", "b")] = "q21"
    transition_table[("q0", "b")] = "q21"
    transition_table[("q21", "r")] = "q22"
    transition_table[("q22", "e")] = "q23"
    transition_table[("q23", "a")] = "q24"
    transition_table[("q24", "d")] = "q25"
    transition_table[("q25", " ")] = "q26"
```

```
    #lexical analysis
    idx_char = 0
    state = 'q0'
    current_token = ''
    while state!='accept':
        current_char = input_string[idx_char]
        current_token += current_char
        state = transition_table[(state, current_char)]
        if state=='q25':
            print('current token: ' , current_token,', valid')
            current_token = ''
        if state== 'error':
            print('eror')
            break;
        idx_char = idx_char + 1
    #conclusion
    if state=='accept':
        print('semua token di input: ', sentence, ', valid')

    return LA
```

Berikut adalah hasil pengujian untuk *lexical analysis* yang digunakan untuk menguji kata masukan yang telah ditentukan. Berikut ada tiga inputan yang berhasil (valid) dengan inputan kata *she*, *likes*, dan *bread*.



Sedangkan dibawah ini adalah hasil pengujian ada tiga inputan yang tidak valid atau tidak berhasil dengan inputan kata *he*, *like*, *breads*. Kata tersebut tidak mengikuti pola yang telah ditentukan sehingga dilaporkan sebagai token tak dikenal atau tidak valid. Untuk kata *breads* sendiri *error* tetapi disini kami melakukan pembacaan kata *bread*, sehingga jika terjadi penambahan huruf akan *error*. Jadi jika terjadi saltik (*typo*) diakhir kata akan ditunjukkan kata yang dapat diterima pada program ini.



Selain itu kami juga mencoba menginput salah satu kata dengan berbagai jumlah spasi. Yang pertama dengan satu spasi, inputan tersebut valid, dengan dua spasi juga valid, serta dengan tiga spasi juga valid. Hal itu membuktikan bahwa spasi berapapun akan tetap valid yang terpenting kata yang diinput sesuai dengan kata yang telah ditentukan pada terminal.

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                3: Python

[redacted]:~/Documents/@SESSION IV/Tba/FinalTask/4/deep$ python3 LA.py
Tugas Besar Teori Bahasa Automata | IF-43-01
Lexical Analyzer
Anindika Riska Intan Fauzy(1301194254)
Bagus Seno Pamungkas(1301190337)
Fauzi Arya Surya Abadi(1301194101)

Terminal: we - she - eats - loves - likes - her - them - cat - it - bread

input in here:  she
current token:   she , valid
semua token di input:   she , valid
[redacted]:~/Documents/@SESSION IV/Tba/FinalTask/4/deep$ python3 LA.py
Tugas Besar Teori Bahasa Automata | IF-43-01
Lexical Analyzer
Anindika Riska Intan Fauzy(1301194254)
Bagus Seno Pamungkas(1301190337)
Fauzi Arya Surya Abadi(1301194101)

Terminal: we - she - eats - loves - likes - her - them - cat - it - bread

input in here:   she
current token:    she , valid
semua token di input:    she , valid
[redacted]:~/Documents/@SESSION IV/Tba/FinalTask/4/deep$ python3 LA.py
Tugas Besar Teori Bahasa Automata | IF-43-01
Lexical Analyzer
Anindika Riska Intan Fauzy(1301194254)
Bagus Seno Pamungkas(1301190337)
Fauzi Arya Surya Abadi(1301194101)

Terminal: we - she - eats - loves - likes - her - them - cat - it - bread

input in here:    she
current token:     she , valid
semua token di input:     she , valid
```

4.2 Parser

Kami menggabungkan program *lexical analyzer* diatas dengan program *parser* dibawah ini menjadi satu program. Program ini hanya meminta satu kali inputan berupa kalimat sepanjang tiga kata sesuai *grammar* yang telah kami tetapkan dengan struktur SB-VB-OB dalam bahasa inggris untuk satu kali menjalankan program. Dibawah ini adalah hasil program gabungan antara *lexical analyzer* dan *parser* yang telah kami rancang.

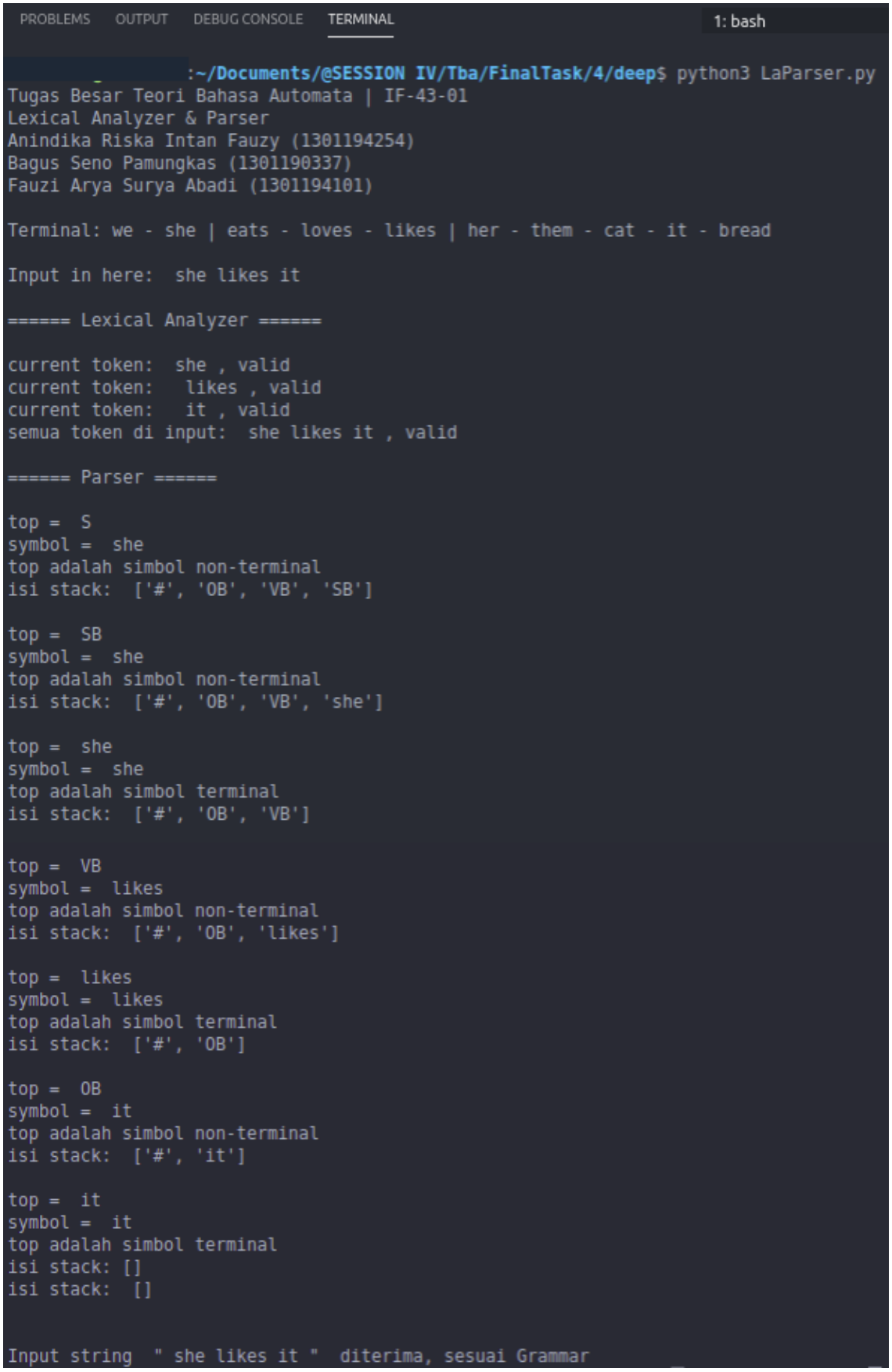
```
def Parser(sentence):
    print("\n====== Parser ====== \n")
    tokens = sentence.lower().split()
    tokens.append('EOS')
    # symbols definition
    non_terminals = ['S','SB','VB','OB']
    terminals = ['we','she','eats','loves','likes',
                 'her','them','cat','it','bread',
                 ]

    parse_table = {}

    parse_table[('S','we')] = ['SB','VB','OB']
    parse_table[('S','she')] = ['SB','VB','OB']
    parse_table[('S','eats')] = ['error']
    parse_table[('S','loves')] = ['error']
    parse_table[('S','likes')] = ['error']
    parse_table[('S','her')] = ['SB','VB','OB']
    parse_table[('S','them')] = ['SB','VB','OB']
    parse_table[('S','cat')] = ['SB','VB','OB']
    parse_table[('S','it')] = ['SB','VB','OB']
    parse_table[('S','bread')] = ['SB','VB','OB']
    parse_table[('S','EOS')] = ['error']

    parse_table[('SB','we')] = ['we']
    parse_table[('SB','she')] = ['she']
    parse_table[('SB','eats')] = ['error']
    parse_table[('SB','loves')] = ['error']
    parse_table[('SB','likes')] = ['error']
    parse_table[('SB','her')] = ['error']
    parse_table[('SB','them')] = ['error']
    parse_table[('SB','cat')] = ['error']
    parse_table[('SB','it')] = ['error']
    parse_table[('SB','bread')] = ['error']
    parse_table[('SB','EOS')] = ['error']

    parse_table[('VB','we')] = ['error']
    parse_table[('VB','she')] = ['error']
    parse_table[('VB','eats')] = ['eats']
    parse_table[('VB','loves')] = ['loves']
    parse_table[('VB','likes')] = ['likes']
    parse_table[('VB','her')] = ['error']
    parse_table[('VB','them')] = ['error']
    parse_table[('VB','cat')] = ['error']
    parse_table[('VB','it')] = ['error']
    parse_table[('VB','bread')] = ['error']
    parse_table[('VB','EOS')] = ['error']

    parse_table[('OB','we')] = ['error']
    parse_table[('OB','she')] = ['error']
    parse_table[('OB','eats')] = ['error']
    parse_table[('OB','loves')] = ['error']
    parse_table[('OB','likes')] = ['error']
    parse_table[('OB','her')] = ['her']
    parse_table[('OB','them')] = ['them']
    parse_table[('OB','cat')] = ['cat']
    parse_table[('OB','it')] = ['it']
    parse_table[('OB','bread')] = ['bread']
    parse_table[('OB','EOS')] = ['error']

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

    #input reading initialization
    idx_token = 0
    symbol = tokens[idx_token]

    #parse table process
    while (len(stack) > 0):
        top = stack[len(stack)-1]
        print('top = ', top)
        print('symbol = ', symbol)
        if top in terminals:
```

```
            print('top adalah simbol terminal')
            if top == symbol:
                stack.pop()
                idx_token = idx_token + 1
                symbol = tokens[idx_token]
                if symbol == "EOS":
                    stack.pop()
                    print('isi stack:', stack)
            else:
                print('error')
                break;
        elif top in non_terminals:
            print('top adalah simbol non-terminal')
            if parse_table[(top, symbol)][0] != 'error':
                stack.pop()
                symbol_to_be_pushed = parse_table[(top, symbol)]
                for i in range(len(symbol_to_be_pushed)-1,-1,-1):
                    stack.append(symbol_to_be_pushed[i])
            else:
                print('error')
                break;
        else:
            print('error')
            break;
        print('isi stack: ', stack)
        print()

    #conlusion
    print()
    if symbol == 'EOS' and len(stack)== 0:
        print('Input string ','"', sentence,'"', ' diterima, sesuai Grammar')
    else:
        print('Error, input string:','"', sentence,'"', ', tidak diterima, tidak sesuai Grammar')

    return Parser
```

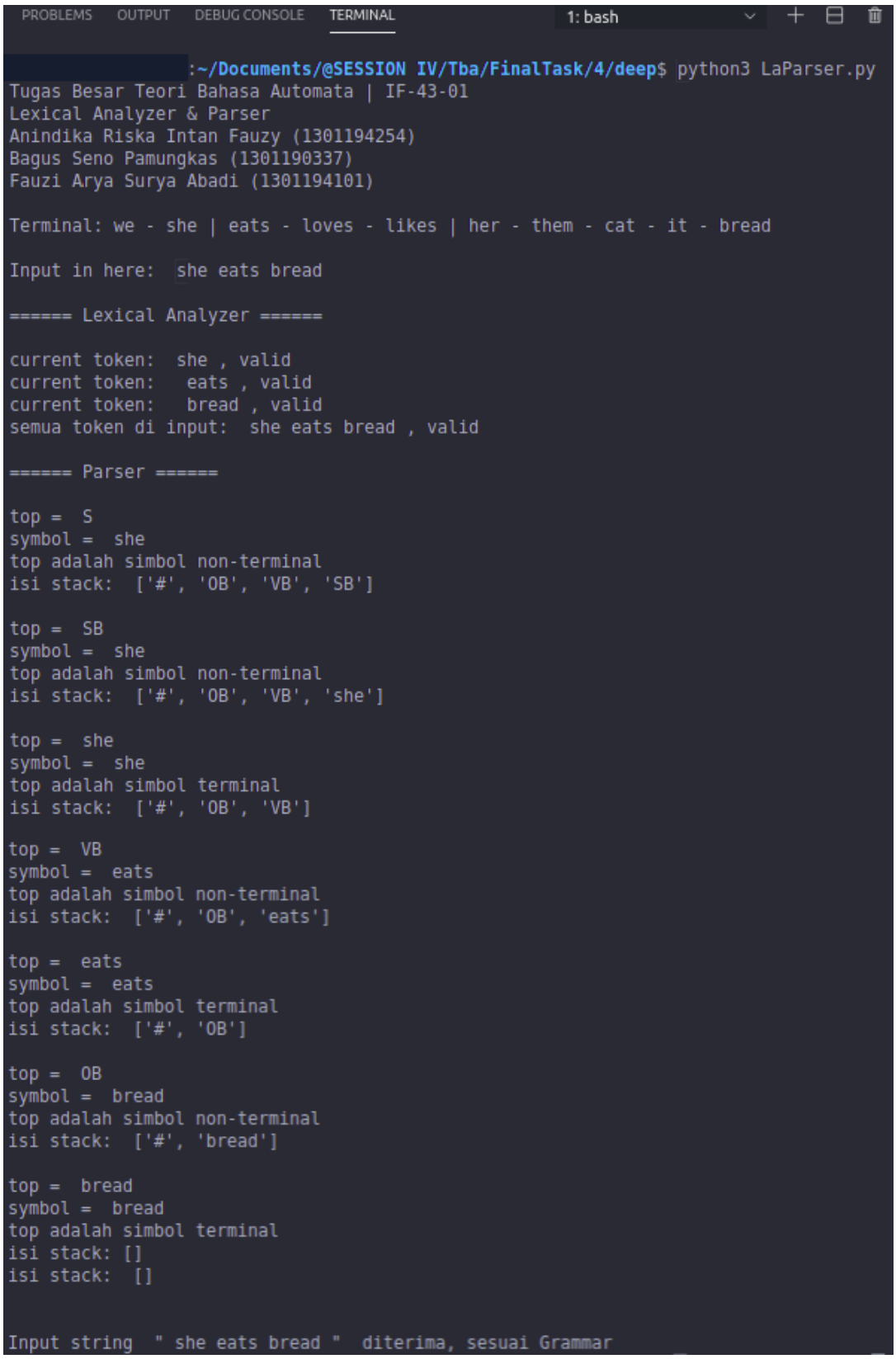
```
print("Terminal: we - she | eats - loves - likes | her - them - cat - it - bread \n")
sentence = input("Input in here:  ")
input_string = sentence.lower()+'#'
LA(sentence)
Parser(sentence)
```

Berikut adalah hasil pengujian program gabungan untuk *lexical analysis* dan *parser* yang digunakan untuk menguji kalimat masukan yang telah ditentukan apakah telah sesuai dengan *grammar* atau tidak. Dibawah ini terdapat tiga inputan kalimat yang telah sesuai dengan *grammar* dengan kata lain inputan tersebut diterima.

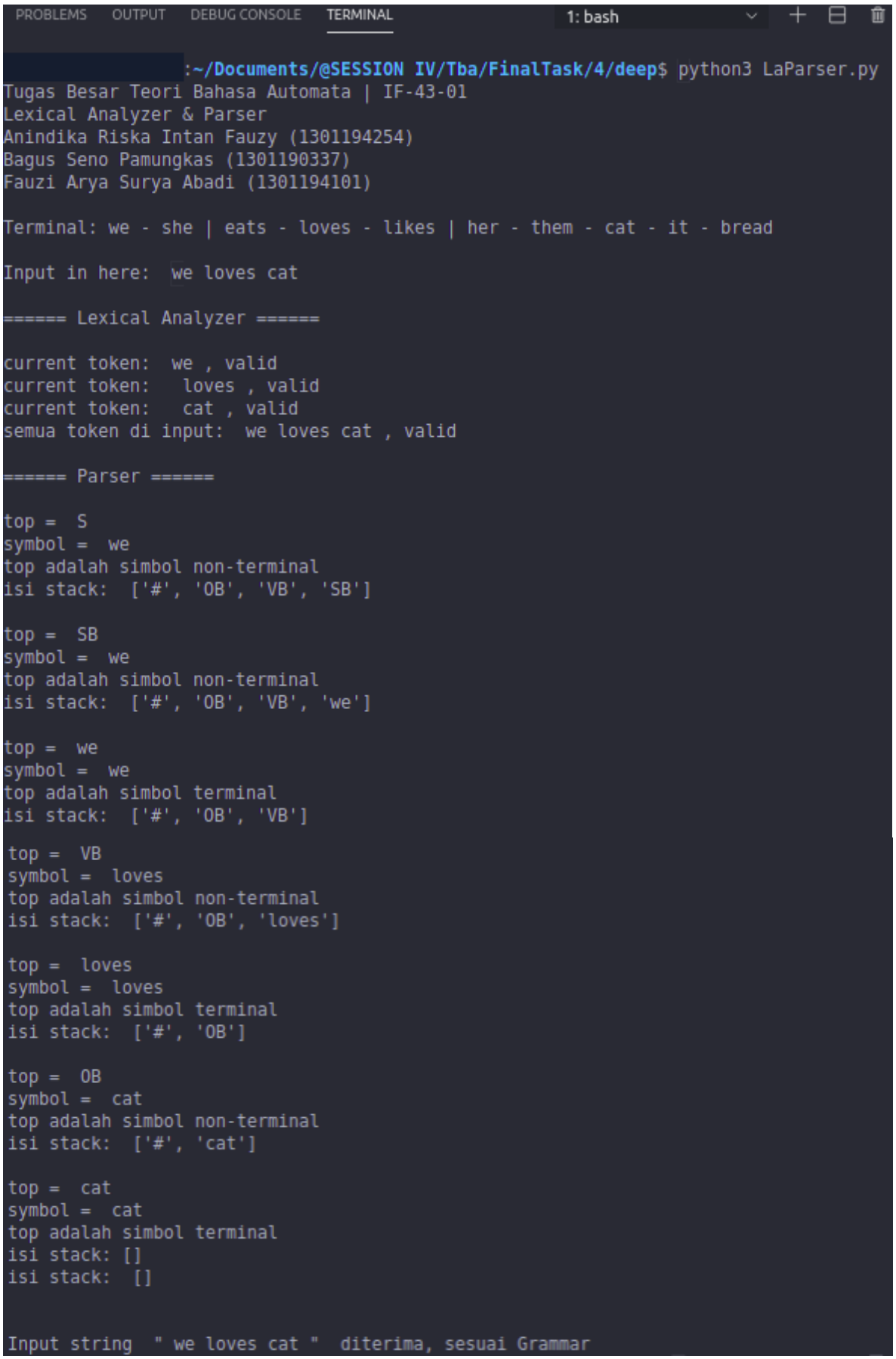
1. she likes it



2. she eats bread

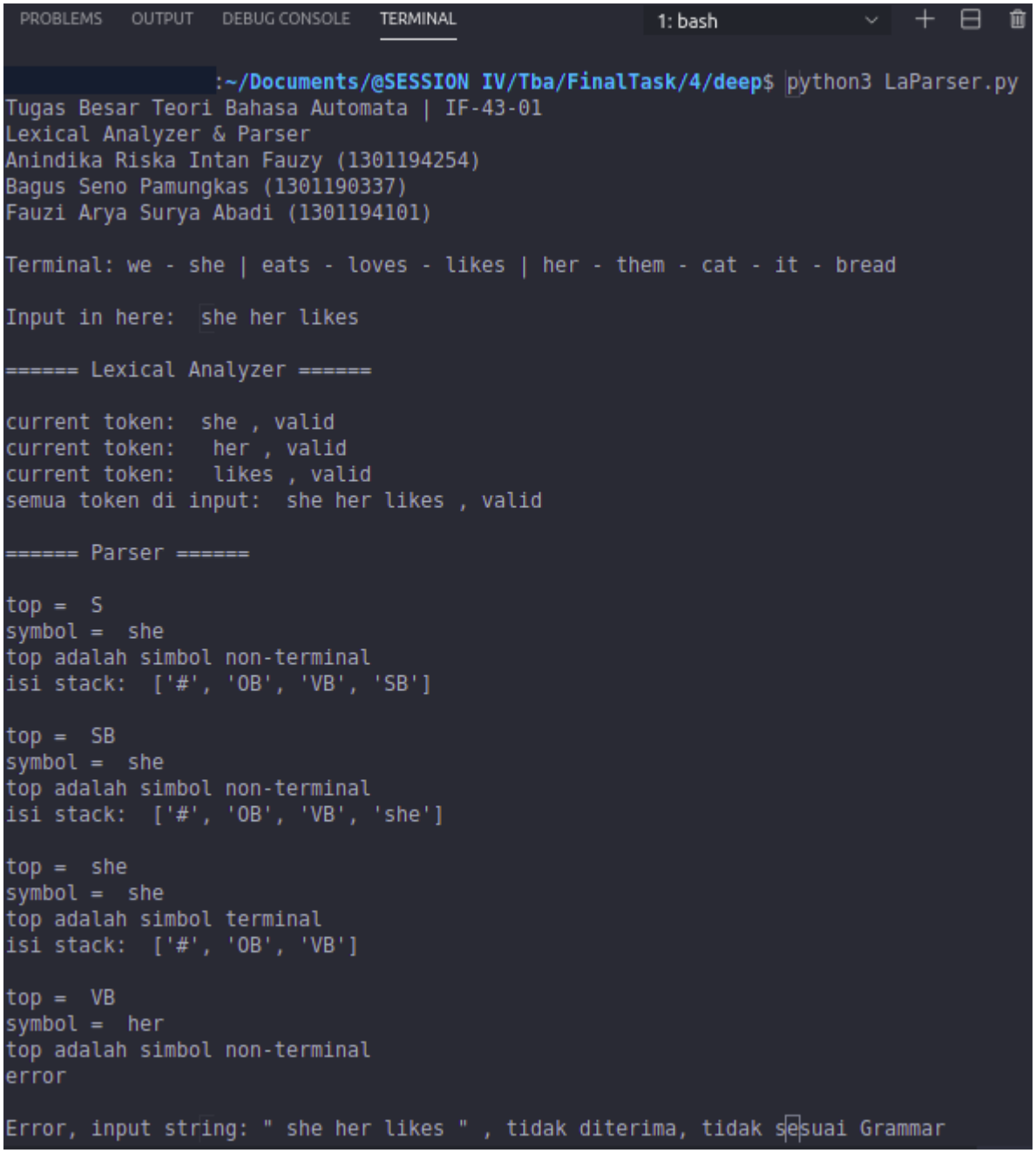


3. we loves cat

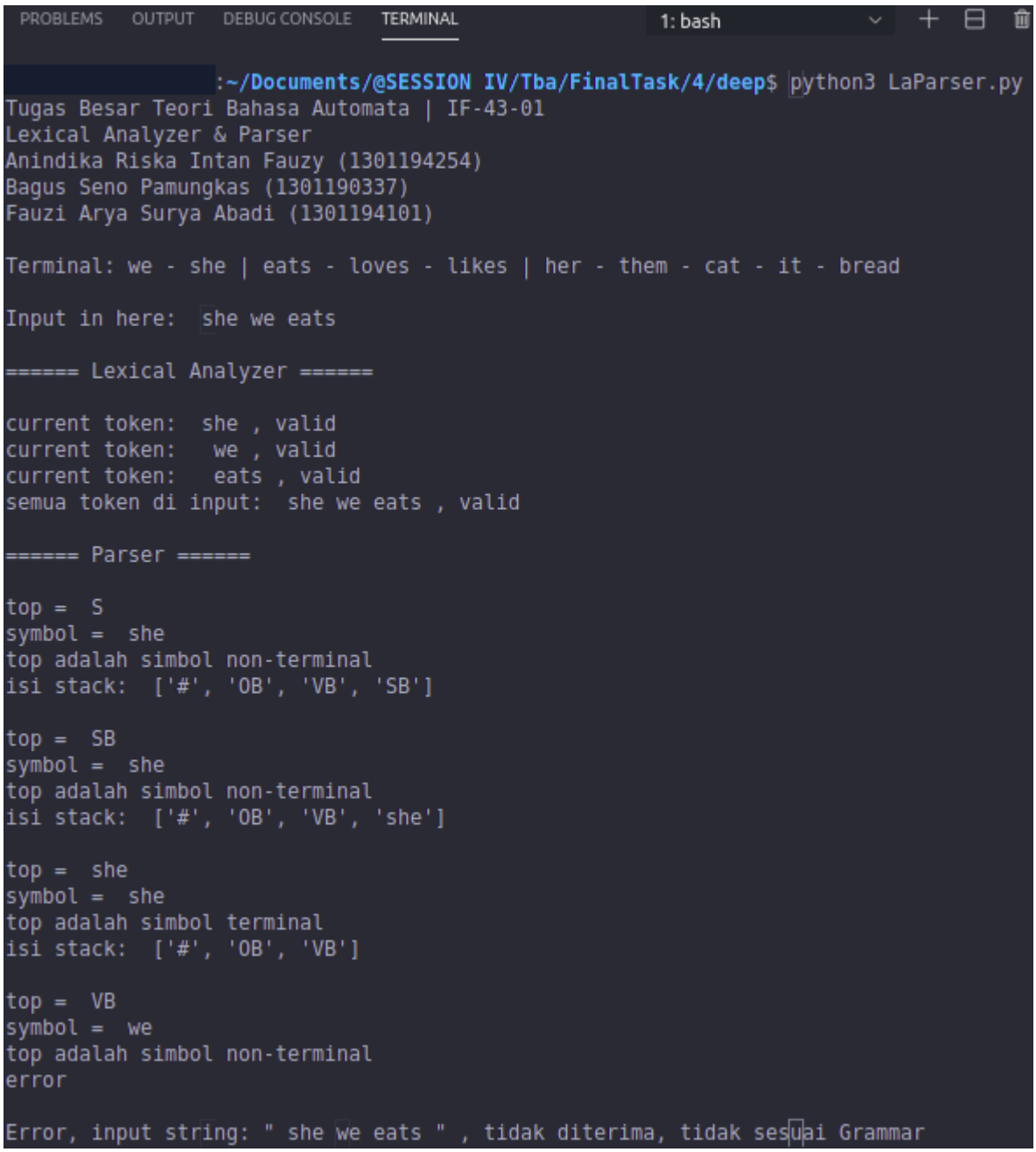


Kami telah menentukan bahwa menggunakan *grammar* (tata bahasa) dengan struktur SB-VB-OB oleh karena itu jika struktur bahasa tersebut terbalik atau tertukar maka akan mempengaruhi ketidaksesuaian *grammar*. Dibawah ini adalah hasil pengujian ada tiga inputan yang tidak sesuai dengan *grammar* dengan kata lain inputan tersebut tidak diterima.

1. she her likes



2. she we eats


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: bash
:~/Documents/@SESSION IV/Tba/FinalTask/4/deep$ python3 LaParser.py
Tugas Besar Teori Bahasa Automata | IF-43-01
Lexical Analyzer & Parser
Anindika Riska Intan Fauzy (1301194254)
Bagus Seno Pamungkas (1301190337)
Fauzi Arya Surya Abadi (1301194101)
Terminal: we - she | eats - loves - likes | her - them - cat - it - bread
Input in here: she we eats
====== Lexical Analyzer ======
current token: she , valid
current token: we , valid
current token: eats , valid
semua token di input: she we eats , valid
====== Parser ======
top = S
symbol = she
top adalah simbol non-terminal
isi stack: ['#', 'OB', 'VB', 'SB']
top = SB
symbol = she
top adalah simbol non-terminal
isi stack: ['#', 'OB', 'VB', 'she']
top = she
symbol = she
top adalah simbol terminal
isi stack: ['#', 'OB', 'VB']
top = VB
symbol = we
top adalah simbol non-terminal
error
Error, input string: " she we eats " , tidak diterima, tidak sesuai Grammar

3. cat loves we

```
                :~/Documents/@SESSION IV/Tba/FinalTask/4/deep$ python3 LaParser.py
Tugas Besar Teori Bahasa Automata | IF-43-01
Lexical Analyzer & Parser
Anindika Riska Intan Fauzy (1301194254)
Bagus Seno Pamungkas (1301190337)
Fauzi Arya Surya Abadi (1301194101)

Terminal: we - she | eats - loves - likes | her - them - cat - it - bread

Input in here:  cat loves we

====== Lexical Analyzer ======

current token:  cat , valid
current token:   loves , valid
current token:   we , valid
semua token di input:  cat loves we , valid

====== Parser ======

top =  S
symbol =  cat
top adalah simbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'SB']

top =  SB
symbol =  cat
top adalah simbol non-terminal
error

Error, input string: " cat loves we " , tidak diterima, tidak sesuai Grammar
```

Daftar Pustaka

1. Ahmad. 2021. “Cara Menulis Daftar Pustaka Dari Buku, Jurnal, Artikel, Website”, https://www.gramedia.com/best-seller/cara-menulis-daftar-pustaka/
2. Collins, Michael. “Context Free Grammars”, http://aritter.github.io/courses/5525_slides/cfg.pdf
3. Aulia, Alvina. 2019. “Teknik Kompilasi: Tahapan Kompilasi”, https://socs.binus.ac.id/2019/12/23/teknik-kompilasi-first-set-pada-top-down-parsing/
4. Aulia, Alvina. 2018. “Penyederhanaan Context Free Grammar”, https://socs.binus.ac.id/2018/12/20/penyederhanaan-context-free-grammar/