LAPORAN TUGAS BESAR

TEORI BAHASA DAN AUTOMATA



I Wayan Ardi Satya Putra - 1301201397

Intan Fauzia Anwar - 1301204539

Muammar Fajar Rahmadani - 1301204129

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY**

**2022**

DAFTAR ISI

[LAPORAN TUGAS BESAR 1](#_Toc106228356)

[TEORI BAHASA DAN AUTOMATA 1](#_Toc106228357)

[DAFTAR ISI 2](#_Toc106228358)

[BAB I 3](#_Toc106228359)

[PENDAHULUAN 3](#_Toc106228360)

[**1.1 Latar Belakang** 3](#_Toc106228361)

[BAB II PENYELESAIAN 4](#_Toc106228362)

[**2.1 Context Free Grammar** 4](#_Toc106228363)

[**2.2 Rancangan Finite Automata** 5](#_Toc106228364)

[a. Diagram Transisi 5](#_Toc106228365)

[b. Tabel Transisi 6](#_Toc106228366)

[**2.3 Tabel Parser** 6](#_Toc106228367)

[BAB III PROGRAM 8](#_Toc106228368)

[**3.1 Lexical analyzer** 8](#_Toc106228369)

[a. Code Program 8](#_Toc106228370)

[b. Input dan Output dari code program 13](#_Toc106228371)

[**3.2 Program Parser** 15](#_Toc106228372)

[a. Code Program 15](#_Toc106228373)

[b. Input dan Output dari code program 18](#_Toc106228374)

[**3.3 Program WEB** 19](#_Toc106228375)

[BAB IV 20](#_Toc106228376)

[KESIMPULAN 20](#_Toc106228377)

BAB I

PENDAHULUAN

**1.1 Latar Belakang**

Manusia merupakan makhluk sosial yang membutuhkan manusia lainnya untuk saling berkomunikasi. Oleh karena itu dibutuhkan suatu bahasa yang dapat dimengerti. Dan bahasa yang digunakan oleh manusia adalah bahasa alami.

Arti dari bahasa alami itu sendiri adalah bahasa yang dapat dikemukakan, ditulis atau diisyaratkan (secara visual atau isyarat lain) oleh manusia untuk berkomunikasi.

Terdapat suatu bahasa yang tidak terdapat pembatasan tata bahasa dalam hasil produksinya*.* Bahasa ini diciptakan oleh para ilmuwan yang terinspirasi dengan adanya bahasa alami, para ilmuwan ini mengembangkan bahasa pemrograman dengan memberikan grammar kedalam bahasa tersebut secara formal, grammar ini diciptakan secara bebas-konteks dan pada akhirnya disebut dengan *Context Free Grammar*.

**BAB II**  
PENYELESAIAN

**2.1 Context Free Grammar**

Pada tugas kali ini, Mahasiswa diminta untuk mendefinisikan suatu *Context Free Grammar* (CFG) yang dapat merepresentasikan bahasa alami atau bahasa yang digunakan oleh manusia ke dalam aturan bahasa sederhana. Dan kelompok kami memilih menggunakan bahasa Banjar dengan struktur S-V-O (*subject-verb-object*).

Untuk membuat program yang sesuai spesifikasi yang dibutuhkan, kami menggunakan *Context Free Gramma*r untuk mempermudah dalam membuat aturan-aturan yang diperlukan. Pada program kali ini, kami menggunakan batasan yang dapat digunakan, dimana batasannya seperti tabel dibawah ini:

|  |  |  |
| --- | --- | --- |
| Subject | Verb | Object |
| Ulun, Ikam, Amang, Wadai, Iwak, Ading, Abah | Maigut, Mamakan, Manatak | Ulun, Ikam, Amang, Wadai, Iwak, Ading, Abah |

G = {V, T, P, S}

V = {NN, VB}

T = {Ulun, Ikam, Amang, Wadai, Iwak, Ading, Abah}

S = {S}

Berikut adalah contoh CGF:

S ⇾ NN VB NN

NN ⇾ Ulun | Ikam | Amang | Wadai | Iwak | Ading | Abah

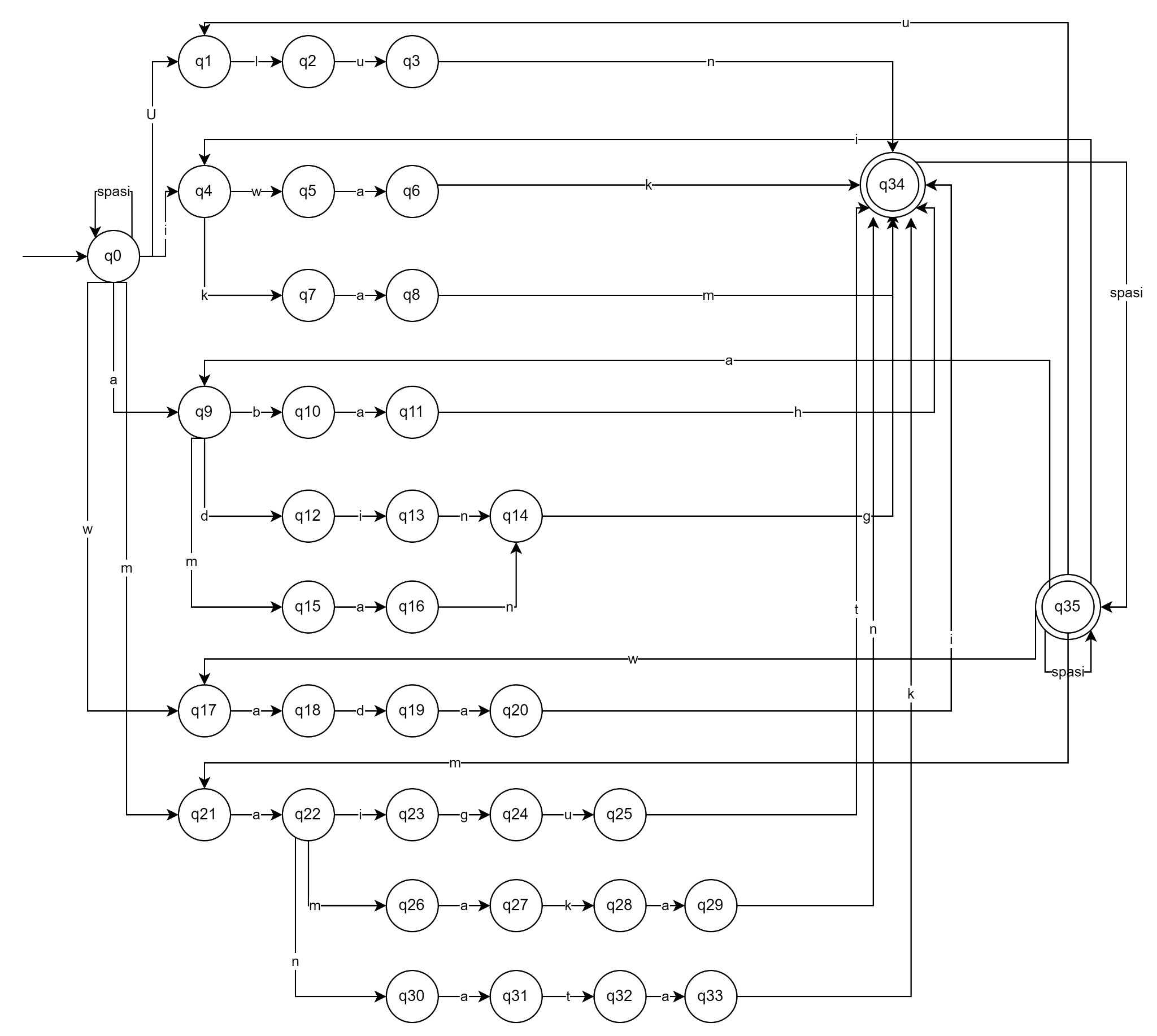
VB ⇾ Maigut | Mamakan | Manatak

**2.2 Rancangan Finite Automata**

Pada rancangan finite automa ini terdapat 35 state dengan q0 sebagai state awal dan 2 accepted state yaitu q34 dan q35. Finite automata ini digunakan sebagai rancangan untuk memvalidasi kata-kata dari tiap huruf dari kata yang sudah ditentukan.

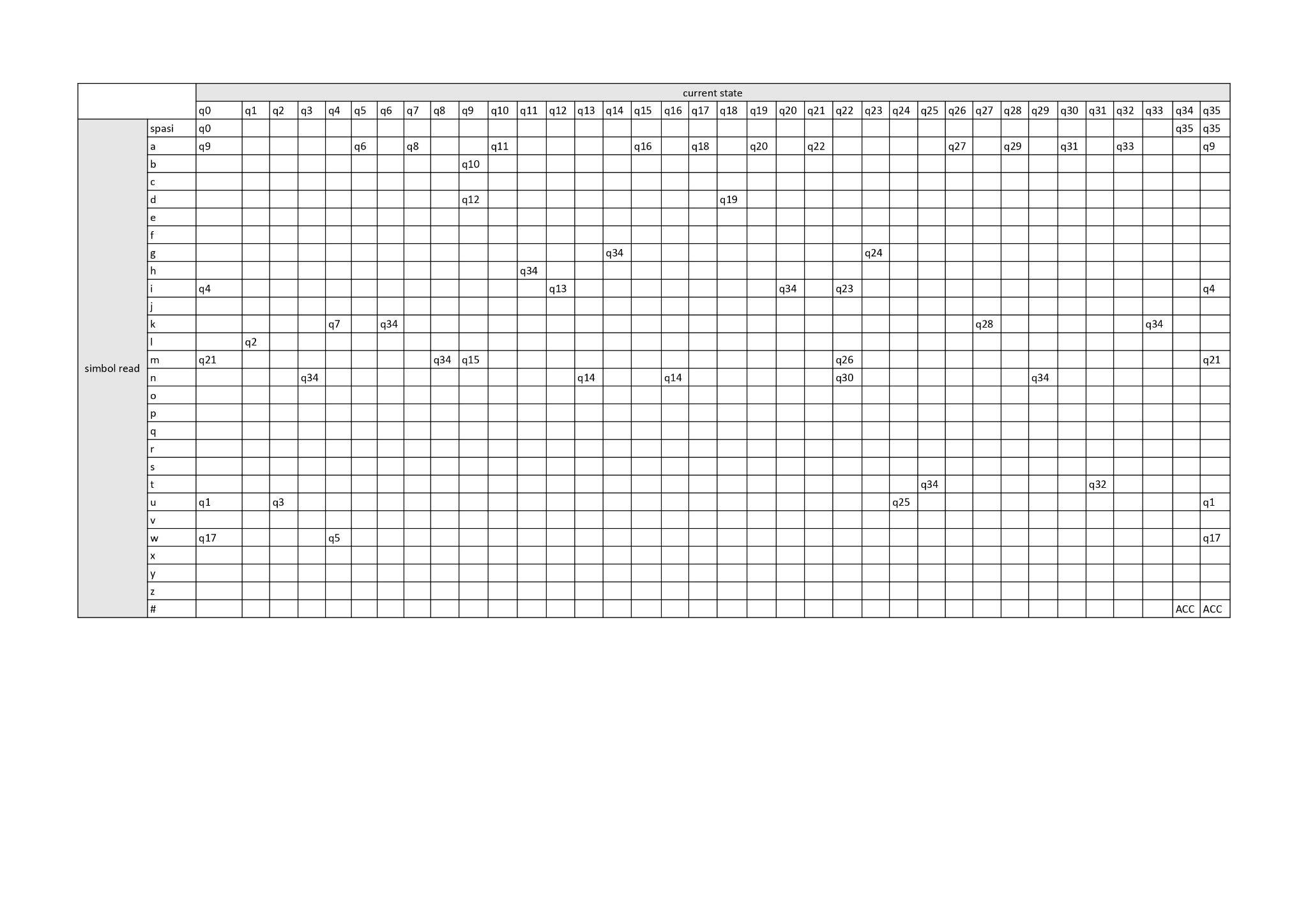
1. Diagram Transisi

Berikut ini adalah rancangan finite automata yang sudah kami buat untuk setiap katanya.

File bisa dilihat [disini](https://app.diagrams.net/#G1Ng4gnTlnggNBNKaq25KHvgfRpIk7ibVQ)

1. Tabel Transisi

Berikut ini adalah tabel transisi yang sudah kami buat untuk setiap katanya.



File bisa dilihat [disini](https://docs.google.com/spreadsheets/d/1Krn7yDjVgCeJLVdTL5V6HDGtA-gwp2C7mVVlMe5tJJ0/edit?usp=sharing)

**2.3 Tabel Parser**

Parse adalah serangkaian perintah pada program dan dipisahkan menjadi komponen yang lebih mudah diproses, yang dianalisis untuk sintaks yang benar dan kemudian dilampirkan ke tag yang menentukan setiap komponen.

Parsing adalah memecah sebuah kalimat atau kelompok kata menjadi komponen yang terpisah, termasuk definisi fungsi atau bentuk setiap bagian yang dapat lebih mudah untuk dimengerti. Pada tugas kali ini, kami akan melakukan parsing dengan menggunakan parse tabel.

Berikut adalah parse table dari kata-kata yang sudah ditentukan sebelumnya oleh kelompok kami:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Ulun** | **Ikam** | **Iwak** | **Abah** | **Ading** | **Amang** | **Wadai** | **Maigut** | **Manatak** | **Mamakan** | **EOS** |
| **S** | **NN VB NN** | **NN VB NN** | **NN VB NN** | **NN VB NN** | **NN VB NN** | **NN VB NN** | **NN VB NN** | **error** | **error** | **error** | **error** |
| **NN** | **ulun** | **ikam** | **iwak** | **abah** | **ading** | **amang** | **wadai** | **error** | **error** | **error** | **error** |
| **VB** | **error** | **error** | **error** | **error** | **error** | **error** | **error** | **maigut** | **manatak** | **mamakan** |  |

BAB III  
PROGRAM

**3.1 Lexical analyzer**

*Lexical analyzer* merupakan program yang digunakan untuk memvalidasi setiap kata yang telah ditentukan. Kelompok kami menggunakan bahasa python sebagai bahasa pemrograman yang akan digunakan untuk membuat program lexical analyzer.

1. Code Program

Berikut adalah program code python lexical analyzer berdasarkan Finite Automata yang telah kami buat sebelumnya :

*#lexical Analyzer*

from pickle import TRUE

import string

*#input str*

*#print("-------| Tubes Teori Bahasa & Automata - Kelompok 11 - IF4412 |-------")*

*#print(" I Wayan Ardi Satya Putra - Intan Fauzia Anwar - Muamar Fajar Rahmadani ")*

*#print(" 1301201397 1301204539 1301204129\n")*

*#print("Terminal Kata : ulun | ikam | amang | wadai | iwak | ading")*

*#print(" abah | maigut | mamakan | manatak ")*

*#kalimat = input("Masukan Kata yang di cari: ")*

*def* lexical(*sentence*):

input\_string = sentence.lower() + "#"

*#initialization*

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", "q27", "q28", "q29", "q30", "q31", "q32",

"q33", "q34","q35"

]

transition\_table = {}

for i in state\_list:

for alphabet in alphabet\_list:

transition\_table[(i, alphabet)] = "ERROR"

transition\_table[(i, "#")] = "ERROR"

transition\_table[(i, " ")] = "ERROR"

*# CFG*

*# S ⇾ NN VB NN*

*# NN ⇾ Ulun | Ikam | Amang | Wadai | Iwak | Ading | Abah*

*# VB ⇾ Maigut | Mamakan | Manatak*

*# For starting node (q0)*

transition\_table[("q0", " ")] = "q0"

*# Finish state*

transition\_table[("q34", "#")] = "ACCEPT"

transition\_table[("q34", " ")] = "q35"

transition\_table[("q35", "#")] = "ACCEPT"

transition\_table[("q35", " ")] = "q35"

*# string "Ulun"*

transition\_table[("q35", "u")] = "q1"

transition\_table[("q0", "u")] = "q1"

transition\_table[("q1", "l")] = "q2"

transition\_table[("q2", "u")] = "q3"

transition\_table[("q3", "n")] = "q34"

*# string "Ikam"*

transition\_table[("q35", "i")] = "q4"

transition\_table[("q0", "i")] = "q4"

transition\_table[("q4", "k")] = "q7"

transition\_table[("q7", "a")] = "q8"

transition\_table[("q8", "m")] = "q34"

*# string "Amang"*

transition\_table[("q35", "a")] = "q9"

transition\_table[("q0", "a")] = "q9"

transition\_table[("q9", "m")] = "q15"

transition\_table[("q15", "a")] = "q16"

transition\_table[("q16", "n")] = "q14"

transition\_table[("q14", "g")] = "q34"

*# string "Wadai"*

transition\_table[("q35", "w")] = "q17"

transition\_table[("q0", "w")] = "q17"

transition\_table[("q17", "a")] = "q18"

transition\_table[("q18", "d")] = "q19"

transition\_table[("q19", "a")] = "q20"

transition\_table[("q20", "i")] = "q34"

*# string "Iwak"*

transition\_table[("q35", "i")] = "q4"

transition\_table[("q0", "i")] = "q4"

transition\_table[("q4", "w")] = "q5"

transition\_table[("q5", "a")] = "q6"

transition\_table[("q6", "k")] = "q34"

*# string "Ading"*

transition\_table[("q35", "a")] = "q9"

transition\_table[("q0", "a")] = "q9"

transition\_table[("q9", "d")] = "q12"

transition\_table[("q12", "i")] = "q13"

transition\_table[("q13", "n")] = "q14"

transition\_table[("q14", "g")] = "q34"

*# string "Abah"*

transition\_table[("q35", "a")] = "q9"

transition\_table[("q0", "a")] = "q9"

transition\_table[("q9", "b")] = "q10"

transition\_table[("q10", "a")] = "q11"

transition\_table[("q11", "h")] = "q34"

*# string "Maigut"*

transition\_table[("q35", "m")] = "q21"

transition\_table[("q0", "m")] = "q21"

transition\_table[("q21", "a")] = "q22"

transition\_table[("q22", "i")] = "q23"

transition\_table[("q23", "g")] = "q24"

transition\_table[("q24", "u")] = "q25"

transition\_table[("q25", "t")] = "q34"

*# string "Mamakan"*

transition\_table[("q35", "m")] = "q21"

transition\_table[("q0", "m")] = "q21"

transition\_table[("q21", "a")] = "q22"

transition\_table[("q22", "m")] = "q26"

transition\_table[("q26", "a")] = "q27"

transition\_table[("q27", "k")] = "q28"

transition\_table[("q28", "a")] = "q29"

transition\_table[("q29", "n")] = "q34"

*# string "Manatak"*

transition\_table[("q35", "m")] = "q21"

transition\_table[("q0", "m")] = "q21"

transition\_table[("q21", "a")] = "q22"

transition\_table[("q22", "n")] = "q30"

transition\_table[("q30", "a")] = "q31"

transition\_table[("q31", "t")] = "q32"

transition\_table[("q32", "a")] = "q33"

transition\_table[("q33", "k")] = "q34"

*# 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 == "q34":

print("current token: {} is valid".format(current\_token))

current\_token = ""

if state == "ERROR":

print("error")

break

idx\_char += 1

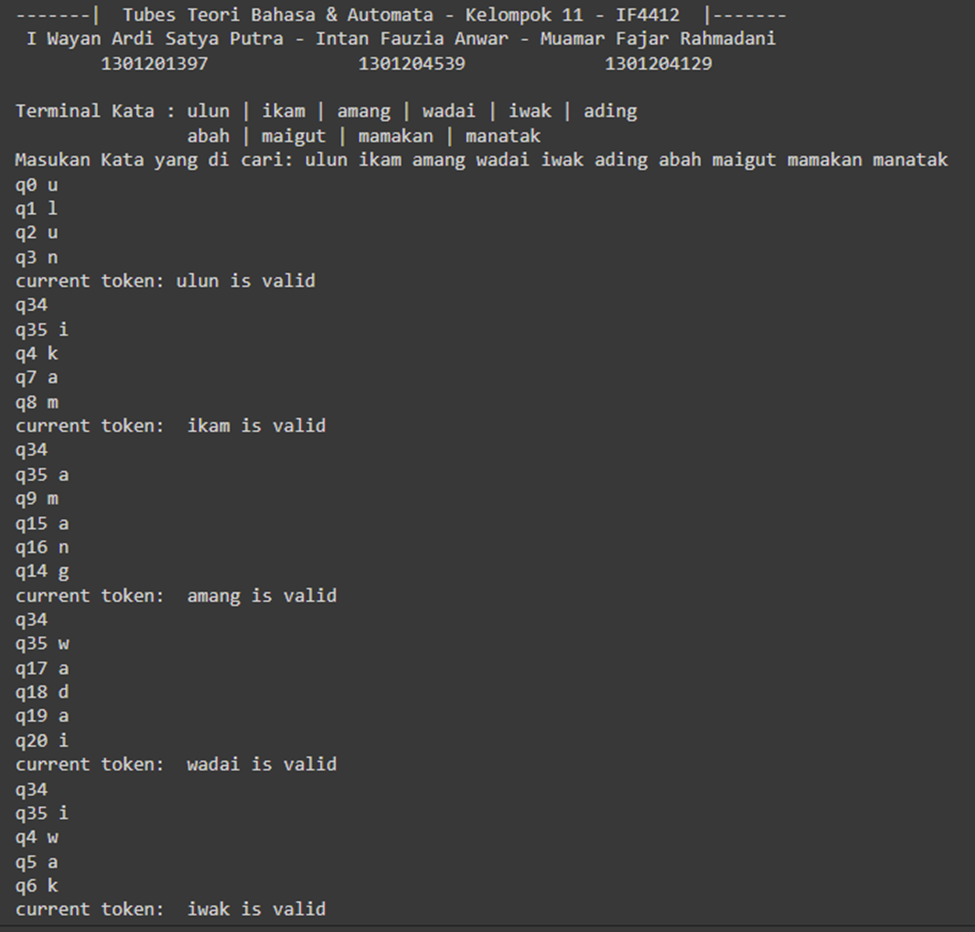
*# Conclusion*

if state == "ACCEPT":

print("semua token yang di input: {} valid".format(input\_string))

return True

1. Input dan Output dari code program

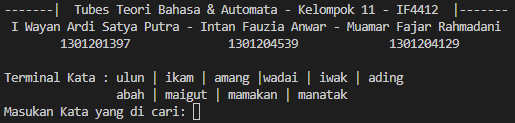


Petunjuk Untuk Menjalankan source code :

1. Lakukan instalasi bahasa pemrograman python
2. Download source code dari link di bawah ini:

<https://colab.research.google.com/drive/1xI2iZlIWgIObsPpaO72EhIDCYxzdYFv5?usp=sharing>

1. Jalankan program tersebut dengan menggunakan python
2. Pada tahap seperti pada gambar di bawah ini, lakukan pengisian kata-kata yang ingin diperiksa oleh lexical analyzer yang telah dibuat oleh kami



1. Jika output yang diberikan program tersebut terdapat ‘error’ maka kata atau kalimat yang anda masukkan tidak valid dengan lexical analyzer yang telah kami rancang.
2. Lalu run program tersebut

**3.2 Program Parser**

Pada program parser ini dilakukan untuk memvalidasi susunan kalimat yang sesuai dengan context free grammar yang sudah dibuat. Program ini akan memberikan keluaran valid dan grammar yang dihasilkan benar apabila susunan inputan kata adalah NN-VB-NN, dimana NN adalah kata benda dan VB adalah kata kerja. Apabila tidak sesuai dengan urutan tersebut program akan memberikan keluaran tidak valid dan grammar tidak diterima. Kami menggunakan Javascript untuk digunakan pada program parser kami.

* 1. Code Program

Berikut ini adalah code program untuk program parser.

***#sentence = input('input kalimat : ')***

***#from tkinter import FALSE, TRUE***

***def* parser(*sentence*):**

**tokens = sentence.lower().split()**

**tokens.append('EOS')**

***#symbol***

**non\_terminals = ['S','NN','VB']**

**terminals=['ulun','ikam','amang','wadai','iwak','ading','abah','maigut','mamakan','manatak']**

***#parse table***

**parse\_table ={}**

**parse\_table[('S', 'ulun')] = ['NN','VB','NN']**

**parse\_table[('S', 'ikam')] = ['NN','VB','NN']**

**parse\_table[('S', 'amang')] = ['NN','VB','NN']**

**parse\_table[('S', 'wadai')] = ['NN','VB','NN']**

**parse\_table[('S', 'iwak')] = ['NN','VB','NN']**

**parse\_table[('S', 'ading')] = ['NN','VB','NN']**

**parse\_table[('S', 'abah')] = ['NN','VB','NN']**

**parse\_table[('S', 'maigut')] = ['error']**

**parse\_table[('S', 'mamakan')] = ['error']**

**parse\_table[('S', 'manatak')] = ['error']**

**parse\_table[('S', 'EOS')] = ['error']**

**parse\_table[('NN', 'ulun')] = ['ulun']**

**parse\_table[('NN', 'ikam')] = ['ikam']**

**parse\_table[('NN', 'amang')] = ['amang']**

**parse\_table[('NN', 'wadai')] = ['wadai']**

**parse\_table[('NN', 'iwak')] = ['iwak']**

**parse\_table[('NN', 'ading')] = ['ading']**

**parse\_table[('NN', 'abah')] = ['abah']**

**parse\_table[('NN', 'maigut')] = ['error']**

**parse\_table[('NN', 'mamakan')] = ['error']**

**parse\_table[('NN', 'manatak')] = ['error']**

**parse\_table[('NN', 'EOS')] = ['error']**

**parse\_table[('VB', 'ulun')] = ['error']**

**parse\_table[('VB', 'ikam')] = ['error']**

**parse\_table[('VB', 'amang')] = ['error']**

**parse\_table[('VB', 'wadai')] = ['error']**

**parse\_table[('VB', 'iwak')] = ['error']**

**parse\_table[('VB', 'ading')] = ['error']**

**parse\_table[('VB', 'abah')] = ['error']**

**parse\_table[('VB', 'maigut')] = ['maigut']**

**parse\_table[('VB', 'mamakan')] = ['mamakan']**

**parse\_table[('VB', 'manatak')] = ['manatak']**

**parse\_table[('NN', 'EOS')] = ['error']**

***# stack initialization***

**stack = []**

**stack.append('#')**

**stack.append('S')**

***# input reading initialization***

**idx\_token = 0**

**symbol = tokens[idx\_token]**

***#***

**while (len(stack) > 0):**

**top = stack [len(stack) - 1]**

**if top in terminals:**

**if top == symbol:**

**stack.pop()**

**idx\_token = idx\_token + 1**

**symbol = tokens[idx\_token]**

**if symbol == 'EOS':**

**stack.pop()**

**else:**

**print('error')**

**break*;***

**elif top in non\_terminals:**

**if parse\_table[(top, symbol)][0] != 'error':**

**stack.pop()**

**symbols\_to\_be\_pushed = parse\_table[(top, symbol)]**

**for i in range(len(symbols\_to\_be\_pushed)-1,-1,-1):**

**stack.append(symbols\_to\_be\_pushed[i])**

**else:**

**print('error')**

**break*;***

**else:**

**print('error')**

**break*;***

**print()**

***# conclusion***

**print()**

**if symbol == 'EOS' and len(stack) == 0:**

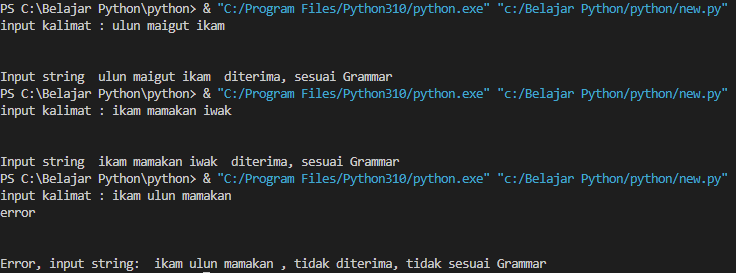
**print('Input string ', sentence, ' diterima, sesuai Grammar')**

**return True**

**else:**

**print('Error, input string: ', sentence, ', tidak diterima, tidak sesuai Grammar')**

**return False**

* 1. Input dan Output dari code program

Petunjuk Untuk Menjalankan source code :

1. Lakukan instalasi bahasa pemrograman python
2. Download source code dari link di bawah ini:

<https://colab.research.google.com/drive/1Cmx4aFNEne2gDyTRddkJbrCFktbfN4aD#scrollTo=-bfyWkuiMQA6>

1. Jalankan program tersebut dengan menggunakan python
2. Pada tahap seperti pada gambar di bawah ini, lakukan pengisian kata-kata yang ingin diperiksa oleh parser yang telah dibuat oleh kami
3. Jika output yang diberikan program tersebut terdapat ‘error’ maka kata atau kalimat yang anda masukkan tidak valid dengan parser yang telah kami rancang.
4. Lalu run program tersebut

**3.3 Program WEB**

Berikut adalah link web dari program Lexical Analyzer dan Parser yang telah kelompok kami buat

<https://lexical-analyzer-parser-kelompok11.netlify.app>

Graphical user interface, website

Description automatically generated

BAB IV

KESIMPULAN

Kesimpulan penyelesaian dari tugas besar yang telah Kami buat dapat disimpulkan bahwa, dengan Bahasa Banjar yang memiliki struktur S-V-O (subject-verb-object) dapat dibuat *context free grammar, finite automata, lexical analyzer, parse table, dan program parser*. Dan dari 5 hal tersebut dapat menunjukkan validasi dari susunan kata dalam Bahasa Banjar