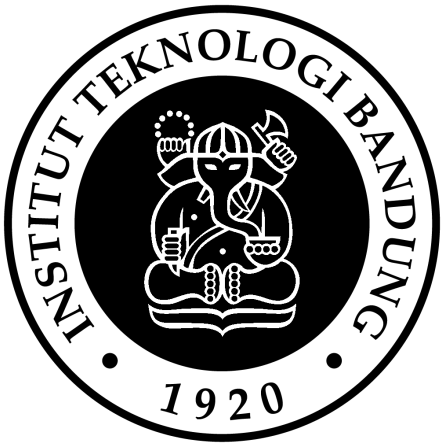
LAPORAN TUGAS BESAR
IF2124/TEORI BAHASA FORMAL DAN AUTOMATA

PYTHON COMPILER



Dipersiapkan oleh :

GForGay

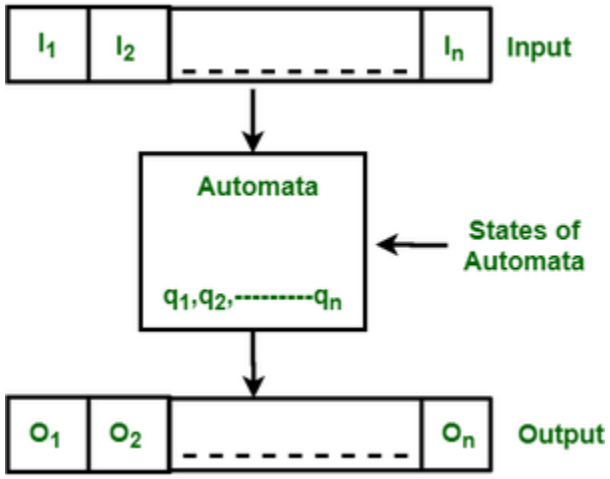
Gibran Darmawan	13520061
Ghebyon Tohada Nainggolan	13520079

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung
Jl. Ganesha no. 10, Bandung 40132

1. TEORI DASAR

1.1 Finite Automata

Finite-automata merupakan model komputasi matematika. Mesin ini merupakan mesi yang abstrak yang memiliki lima elemen atau *tuple.* Mesin ini memiliki sebuah kumpulan *state* dan *rules* untuk berpindah dari satu *state* ke *state* selanjutnya yang bergantung kepada jenis atau simbol input.



FA memiliki berikut jenis simbol atau *tuple*:

```
Q : Finite set of states.
Σ : set of Input Symbols.
    q : Initial state.
F : set of Final States.
δ : Transition Function.
```

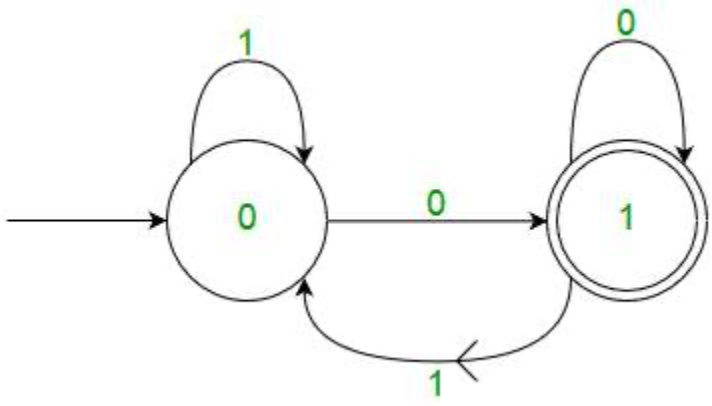
FA (*finite automata*) memiliki dua jenis, yaitu:

1. DFA (*deterministic finite automata)*

   Dalam DFA, untuk setiap input mesin hanya masuk ke satu *state* saja. Fungsi transisi sudah didefinisikan untuk setiap input dalam setiap state. Sebuah DFA tidak bisa berubah state apabila tidak memiliki sebuah input. Dengan aturan di atas, DFA memiliki fungsi transisi berikut:

$$\delta : Q \times \Sigma \dashrightarrow Q$$

Berikut merupakan contoh DFA Σ = {0, 1} yang menerima input string berakhiran dengan 0:

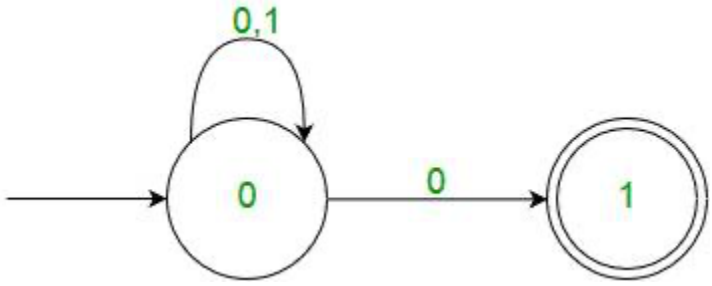


2. NFA (*nondeterministic finite automata)*

   NFA mirip dengan DFA dengan perbedaan bahwa NFA dapat menerima input kosong dan satu input bisa merubah ke banyak state. Dengan perbedaan ini, NFA memiliki fungsi transisi yang berbeda, yaitu:

$$\delta: \quad Q \times (\Sigma \cup \varepsilon) \dashrightarrow 2^{Q}$$

   Berikut contoh sebuah mesin NFA:



1.2 Context Free Grammar

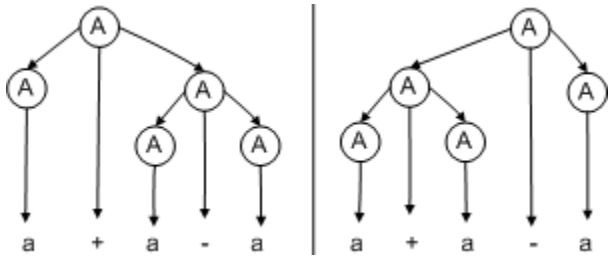
Context free grammar (CFG) merupakan tata bahasa yang mempunyai tujuan sama seperti sebuah bahasa biasa yaitu dengan suatu cara bisa menunjukkan bagaimana menghasilkan suatu untai-untai dalam sebuah bahasa.
CFG mempunyai sebuah definisi formal, yaitu:

G =(V, T, P, S)

V = himpunan terbatas variabel
T = himpunan terbatas terminal
P = himpunan terbatas dari produksi
S = *start symbol*

CFG dapat direpresentasikan dengan pohon penurunan. Nodes dari pohon tersebut melambangkan simbol-simbol dan ujung pohon melambangkan *production rules.* Dedaunan dari pohon tersebut merupakan hasil akhir (*terminal symbols*) yang mengandung string grammar tersebut dengan urutan simbol dan *production rules.* Pohon penurunan di bawah menunjukkan dua cara untuk menghasilkan string ‘a + a - a’ dengan grammar:

$$A \rightarrow A+A \mid A-A \mid a.$$



Karena grammar ini bisa diimplementasikan dengan beberapa pohon turunan untuk menghasilkan string yang sama, hal ini bisa disebut sebagai **ambigu.**

1.3 Chomsky Normal Form

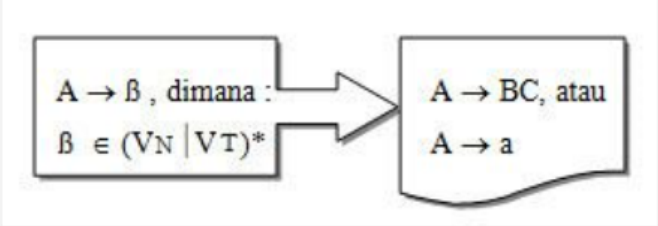
Bentuk normal Chomsky / *Chomsky Normal Form* (CNF) merupakan salah satu bentuk normal yang berguna untuk CFG. CNF dapat dibuat dari sebuah tata bahasa bebas konteks yang telah mengalami penyederhanaan yaitu penghapusan produksi *useless*, unit, dan *ε.* Suatu tata bahasa bebas konteks dapat dibuat menjadi bentuk normal Chomsky dengan syarat:

- Tidak memiliki produksi *useless*
- Tidak memiliki produksi *unit*
- Tidak memiliki produksi *ε*

Bentuk normal Chomsky adalah CFG, dengan setiap produksinya berbentuk :

A → BC atau A → a

Transformasi CFG ke CNF



Aturan produksi dalam bentuk normal Chomsky ruas kanannya tepat berupa sebuah terminal atau dua variabel.

Pembentukan CNF

- Biarkan aturan produksi yang sudah dalam *bentuk normal Chomsky*
- Lakukan penggantian aturan produksi yang ruas kanannya memuat simbol terminal dan panjang ruas kanan > 1
- Lakukan penggantian aturan produksi yang ruas kanannya memuat > 2 simbol variabel
- Penggantian-penggantian tersebut bisa dilakukan berkali-kali sampai akhirnya semua aturan produksi dalam *bentuk normal Chomsky*
- Selama dilakukan penggantian, kemungkinan kita akan memperoleh aturan-aturan produksi baru, dan juga memunculkan simbol-simbol variabel baru.

1.4 Algoritma Cocke Younger Kasami

CYK Algorithm merupakan salah satu algoritma parsing untuk CFG yang seudah diubah ke bentuk CNF. fungsi algoritma ini merupakan *membership testing*, atau menunjukkan apa sebuah string dapat diterima atau bagian dari bahasa CFG tersebut.

Contoh

Misal grammar G adalah

```
S -> AB | BC
A -> BA | a
B -> CC | b
C -> AB | a
```

Buktikan bahwa **baaba** ada dalam L(G)

1. Masukkan *single length rule* ke tabel di bawah

	b	**a**	**a**	**b**	**a**
b	{B}				
a		{A,C}			
a			{A,C}		
b				{B}	
a					{A,C}

2. Isi sisa tabel

	b	**a**	**a**	**b**	**a**
b	{B}	{S,A}	Φ	Φ	{S,A,C}
a		{A,C}	{B}	{B}	{S,A,C}
a			{A,C}	{S,C}	{B}
b				{B}	{S,A}
a					{A,C}

3. Perhatikan bahwa S ada dalam sel tabel {1,5}, maka string **baaba** termasuk dalam L(G).

1.5 Bahasa Pemrograman Python

Python adalah bahasa pemrograman interpretatif multiguna. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Hal ini membuat Python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain.

Bahasa ini muncul pertama kali pada tahun 1991, dirancang oleh seorang bernama Guido van Rossum. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation. Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi. Saat ini kode python dapat dijalankan di berbagai platform sistem operasi, beberapa di antaranya adalah Linux/Unix, Windows, Mac OS/X, Java Virtual Machine, Amiga, Palm, Symbian.

Python menggunakan duck typing dan memiliki objek yang diketik tetapi nama variabel yang tidak diketik. Batasan jenis tidak diperiksa pada waktu kompilasi; sebaliknya, operasi pada suatu objek mungkin gagal, menandakan bahwa objek yang diberikan bukan tipe yang sesuai. Meskipun diketik secara dinamis, Python diketik dengan kuat, melarang operasi yang tidak terdefinisi dengan baik (misalnya, menambahkan angka ke string) daripada secara diam-diam mencoba memahaminya.

Dengan program yang tubes ini buat, hampir semua hal dalam paragraf sebelumnya bisa terwujudkan.

2. Hasil

2.1 Hasil Regular Expression

Regex in Python	Representasi	Digunakan
r'[_a-zA-Z]+[_A-Za-z0-9]*'	variabel	Ya
r'^[-+]?[0-9]+$'	integer	Ya
r'[+-]?[0-9]+\.[0-9]+'	float	Ya
r'[#].*'	Comment one line	Tidak
r'\'\'\'[\S\s]*\'\'\''	Comment multiline	Tidak
r'\"\"\"[\S\s]*\"\"\"'	Comment multiline	Tidak
r'\'[\w\W]*\'' : "string"	string	Tidak
r'\"[\w\W]*\"' : "string"	string	Tidak

2.2 Hasil CFG

Definisi formal CFG : **G =(V, T, P, S)**

Start Symbol (S) : S

Variabel/Non-Terminal Symbol (V)

S	PRINT	WHILE	BINOP	INBRACKET
VAL	IF	DEF	UNOP	ASSIGNMENT
VAR	ELIF	CLASS	METHOD	TYPE
SVAL	ELSE	IMPORT	STRING	COMPARATION
INPUT	FOR	CONTENT	BOOL	EXPRES
RANGE	RETURN	RAISE	FROM	EXPRES2
IF2	ELIF2	ELSE2	SDEF	CALLFUNCTION

Terminal Symbol(T)

:	`	"	'''	"""	+=	-=	*=
None	string	integer	float	,	#	variable	/=
input	print	(	)	str	int	double	=
len	%	//	**	/	*	-	+
with	open	as	content	True	False	and	or
!=	<=	>=	==	<	>	is	not
range	if	elif	else	for	in	while	def
return	continue	pass	break	raise	from	import	class

Production Symbol (P)

Variable	**Production**
S	S S | ASSIGNMENT | VAR = SVAL | VAR = INPUT | SVAL COMPARATION SVAL | SVAL | PRINT | IF | FOR | WHILE | DEF | CLASS | IMPORT | CONTENT
ASSIGNMENT	VAR += SVAL | VAR -= SVAL | VAR *= SVAL | VAR /= SVAL
VAR	variable
SVAL	VAR | SVAL BINOP SVAL | SVAL , SVAL | METHOD | STRING | float | integer | CONTENT | BOOL | INBRACKET | None | VAL | CALLFUNCTION
CALLFUNCTION	variable INBRACKET
VAL	UNOP SVAL
INPUT	input INBRACKET | TYPE (INPUT)
PRINT	print INBRACKET
INBRACKET	(SVAL) | ( ) | (SVAL FOR)
TYPE	str | float | int | double

BINOP	+ | - | * | / | ** | // | %
UNOP	+ | -
STRING	" string " | ' string '
METHOD	len INBRACKET | with open INBRACKET as VAR | with open INBRACKET as VAR :
CONTENT	''' content ''' | """ content """ | # content
BOOL	True | False | BOOL BOOL | BOOL and BOOL | BOOL or BOOL | not BOOL | SVAL COMPARATION SVAL | SVAL is SVAL
COMPARATION	> | < | == | != | <= | >=
EXPRES	(BOOL) : S | BOOL : S | BOOL : | (BOOL) :
EXPRES2	BOOL) : S | BOOL : S | BOOL : | (BOOL) : | (BOOL) : S RETURN | BOOL : S RETURN | (BOOL) : RETURN | BOOL : RETURN
RANGE	range INBRACKET
IF	if EXPRES | IF ELIF | IF ELSE | IF RAISE | IF BREAK | IF PASS | IF CONTINUE
IF2	if EXPRES2 | IF2 ELIF2 | IF2 ELSE2 | IF2 RAISE | IF2 BREAK | IF2 PASS | IF2 CONTINUE
ELIF	elif EXPRES | ELIF ELIF | ELIF ELSE
ELIF2	elif EXPRES2 | ELIF2 ELIF2 | ELIF2 ELSE2
ELSE	else : S
ELSE2	else : S | else : S RETURN | else : RETURN
FOR	for VAR in RANGE : S | for VAR in INBRACKET : S| for VAR in VAR : S | for VAR in STRING : S
WHILE	while EXPRES | while INBRACKET :
DEF	def VAR INBRACKET : | DEF SDEF
SDEF	S | SDEF SDEF | IF2 | RETURN
CLASS	class VAR : S

IMPORT	import VAR | import VAR as VAR
FROM	from VAR IMPORT
RAISE	raise INBRACKET | raise VAR INBRACKET
BREAK	break
PASS	pass
CONTINUE	continue
RETURN	return BOOL | return SVAL

3. Implementasi dan Pengujian

3.1 Implementasi

3.1.1 File CFG2CNF.py dan File helper.py

File CFG2CNF.py dan File helper.py dikutip dari https://github.com/adelmassimo/CFG2CNF. Kegunaan utama dari program tersebut adalah untuk mengkonversi Context Free Grammar menjadi bentuk Chomsky Normal Form. CFG di load dari file CFG.txt dan dituliskan hasil CNF nya ke file CNF.txt
Yang perlu diperhatikan saat menulis CFG di file CFG.txt:

- Gunakan spasi antar simbol, satu spasi, tidak lebih
- gunakan ';' karakter untuk memisahkan baris dalam produksi: jangan gunakan untuk yang terakhir.
- Tuliskan terminal dan variabel di bagian atas file

Yang perlu diperhatikan saat hendak menggunakan CNF.txt:

- Hapus terlebih dahulu line kosong yang berada di paling bawah di file CNF.txt. Bila tidak dihapus, maka akan terjadi error

Fungsi / Prosedur	Kegunaan
`isUnitary`	Mengecek apakah suatu rules sudah memiliki productions tepat satu buah simbol terminal dan nonterminal menjadi bagian dari daftar variabel
`isSimple`	Mengecek apakah suatu rules sudah simple (memiliki produksi satu buah terminal atau 2 buah variabel)
`START`	Menambahkan suatu start simbol baru yaitu S0
`TERM`	Menghapus rules yang mengandung terminal dan non-terminal simbol sekaligus dan mengubahnya ke dalam bentuk yang simple/CNF(memiliki produksi satu buah terminal atau 2 buah variabel)
`BIN`	Prosedur untuk mengeliminasi non-unitry rules
`DEL`	Prosedur untuk melakukan penghapusan non-terminal rules
`unit_routine`	Memeriksa apakah suatu unit atau rules sudah berbentuk unary atau single
`UNIT`	Mengeliminasi unit production dalam suatu rules

convertToMap	Melakukan konversi dari production yang berbentuk list of list menjadi sebuah map
union	Penggabuang dua list
loadModel	Me-load model dari CFG dan dibagi berdasarkan Terminal, Production, dan Variable
cleanProduction	Melakukan pembersihan pada production dan dimuat dalam bentuk list agar dapat diproses lebih lanjut
cleanAlphabet	Melakukan pembersihan untuk memuat terminal dan variabel dalam list
seekAndDestroy	Melakukan eliminasi useless variabel
setUpDict	Melakukan penelusuran dari unit variabel

3.1.2 File createToken.py

File tersebut berisi fungsi dan prosedur yang mendukung pembuatan token dari bahasa yang diterima sehingga bisa dipakai saat pengecekan dengan menggunakan algoritma CYK.

Fungsi / Prosedur	Kegunaan
concatString1	Fungsi yang bekerja seperti DFA untuk mengubah seluruh karakter yang berada diantara tanda “ ‘ ” menjadi string.
concatString2	Fungsi yang bekerja seperti DFA untuk mengubah seluruh karakter yang berada diantara tanda ‘ “ ‘ menjadi string.
concatMultiLineComment1	Fungsi yang bekerja seperti DFA untuk mengubah seluruh karakter yang berada diantara tanda “ ‘” ” menjadi content
concatMultiLineComment2	Fungsi yang bekerja seperti DFA untuk mengubah seluruh karakter yang berada diantara tanda ‘ “”” ’ menjadi content
concatSingleLineComment	Fungsi yang bekerja seperti DFA untuk mengubah seluruh karakter yang berada diantara tanda ‘ # ’ dan ‘\n’ menjadi content. ‘ \n ’ ikut menjadi content
concatComment	Memuat fungsi concatMultiLineComment1, concatMultiLineComment2, dan concatSingleLineComment
concatString	Memuat fungsi concatString1 dan concatString2

`createToken`	Fungsi untuk membentuk token utuh dari kalimat

3.1.3 File CYK.py

File tersebut berisi fungsi dan prosedur yang berguna dalam pembuatan tabel CYK.

Fungsi / Prosedur	Kegunaan
`LoadCNF`	Prosedur untuk membentuk list of list of tuple. Tuple yang terbentuk terdiri dari 2 elemen. Elemen yang pertama ialah Variabel dan Elemen yang kedua ialah List yang berisi Production dari elemen pertama tuple. Data disimpan ke dalam variabel global agar CNF tersebut dapat diakses langsung oleh fungsi lain
`checkInCNFList`	Fungsi untuk mengembalikan seluruh variabel yang dapat membentuk production yang diminta (*parameter fungsi*)
`combine`	Fungsi pendukung untuk membuat production yang didapatkan dari algoritma CYK. Pada algoritma CYK kita membutuhkan kombinasi beberapa variabel, fungsi inilah yang mengkombinasikannya
`CYK`	Fungsi yang menerapkan algoritma CYK pada token yang sudah dibentuk, dan akan mengembalikan tabel CYK yang terbentuk
`isSyntaxValid`	Fungsi untuk melakukan pengecekan apakah elemen terakhir pada tabelCYK memuat startSymbol. Jika memuat startSymbol, maka token/*language* diterima.

3.1.4 File parserprogram.py

File parserprogram.py adalah file main dari keseluruhan program tersebut. Untuk menjalankan program masukkan command berikut :

```
python parserprogram.py <nama_source_code>
```

3.2 Pengujian

3.2.1 Input/output

Source Code :

TestCase0.txt

```
print("Hello world!")
print("Kami dari kelompok APA")
print("Dari Kelas K", 2)

w = float(input("Masukkan nilai float : "))
x = int(input("Masukkan nilai integer : "))
y = input("Masukkan string : ")
z = str(input("Masukkan string : "))
```

Output :

```
PS D:\Kuliah\Tingkat 2\IF2124 - Teori Bahasa Formal dan Otomata\
Tugas\TUBES_TBFO> python parserprogram.py TestCase0.txt

Accepted
Time Execution:  1.2148914000135846 sec
```

3.2.2 Percabangan

Source Code :

TestCase1.txt

```
if (a == b):
    i += 1
    print("a dan b kok bisa sama")
elif (a > b):
    j += 2
    print("a kok bisa lebih besar dari b")
else:
    print("hah... kok ga bisa dibandingkan?")

if not False:
    raise AssertionError(message)
```

Output :

```
PS D:\Kuliah\Tingkat 2\IF2124 - Teori Bahasa Formal dan Otomata\Tug
as\TUBES_TBFO> python parserprogram.py TestCase1.txt

Accepted
Time Execution:  0.10532090003835037 sec
```

3.2.3 For Loop

Source Code :

TestCase2.txt

```
for i in x:
    print(i)

for j in range(10):
    print("Saya Satpol")

for k in range(10,-1,-1):
    print(k)

for l in "UwU":
    print(l)
```

Output :

```
PS D:\Kuliah\Tingkat 2\IF2124 - Teori Bahasa Formal dan Otomata
\Tugas\TUBES_TBFO> python parserprogram.py TestCase2.txt

Accepted
Time Execution:  0.3034343000035733 sec
```

3.2.4 While Loop

Source Code :

TestCase3.txt

```
while (myLove == True):
    print("Saranghae")
    if (weBrokeUp == True):
        print("stop it !!!")
        myLove = False
```

Output :

```
PS D:\Kuliah\Tingkat 2\IF2124 - Teori Bahasa Formal dan Otomata\Tugas\
TUBES_TBFO> python parserprogram.py TestCase3.txt

Accepted
Time Execution:  0.9537259000353515 sec
```

3.2.5 Class

Source Code :

TestCase4.txt

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

Output :

```
PS D:\Kuliah\Tingkat 2\IF2124 - Teori Bahasa Formal dan Otomata\Tugas
\TUBES_TBFO> python parserprogram.py TestCase4.txt

Accepted
Time Execution:  0.15523469995241612 sec
```

3.2.6 Fungsi dan Prosedur

Source Code :

TestCase5.txt

```
def persentase (total, jumlah):
    if (total >= 0 and total <= jumlah):
        return total / jumlah * 100
    return False

def daftarMenu():
    print('''
    1. Ayam Goreng
    2. Ayam Madu
    3. Ayam ga ada rasa
    4. Ayam ga ada otak
    5. Bebek terkucilkan
    ''')
```

Output :

```
PS D:\Kuliah\Tingkat 2\IF2124 - Teori Bahasa Formal dan Otomata
\Tugas\TUBES_TBFO> python parserprogram.py TestCase5.txt

Accepted
Time Execution:  0.4044088000082411 sec
```

3.2.7 With

Source Code :

TestCase6.txt

```
1   with open('example.txt', 'w') as my_file:
2       my_file.write('Hello world!')
```

Output :

```
PS D:\Kuliah\Tingkat 2\IF2124 - Teori Bahasa Formal dan Otomata\
Tugas\TUBES_TBFO> python parserprogram.py TestCase4.txt

Accepted
Time Execution:  0.14846830000169575 sec
```

3.2.8 Contoh Accepted dari Spek Tubes

Source Code :

TestCaseAcc.txt

```
1   def do_something(x):
2       ''' This is a sample multiline comment
3       '''
4       if x == 0:
5           return 0
6       elif True:
7           if True:
8               return 3
9           else:
10              return 2
11      elif x == 32:
12          return 4
13      else:
14          return "Doodoo"
```

Output :

```
PS D:\Kuliah\Tingkat 2\IF2124 - Teori Bahasa Formal dan Otomata\
Tugas\TUBES_TBFO> python .\parserprogram.py TestCaseAcc.txt

Accepted
Time Execution:  0.3776936999638565 sec
```

3.2.9 Contoh Accepted dari Spek Tubes

Source Code :

TestCaseReject.txt

```
def do_something(x):
    ''' This is a sample multiline comment
    '''
    x + 2 = 3
    if x == 0 + 1
        return 0
    elif x + 4 == 1:
        else:
            return 2
    elif x == 32:
        return 4
    else:
        return "Doodoo"
```

Output :

```
PS D:\Kuliah\Tingkat 2\IF2124 - Teori Bahasa Formal dan Otomata\
Tugas\TUBES_TBFO> python .\parserprogram.py TestCaseReject.txt

Syntax error!
Time Execution:  0.15593619999708608 sec
```

3.2.10 Comment Single dan Multi Line

Source Code :

TestCase8.txt

```
# Ini adalah comment dengan single line \n
''' Ini adalah comment dengan multi line
Apakah bisa?
Tentu saja bisa
'''
""" Ini juga comment dengan multi line
Ya Tuhan semoga bisa
Ah berserah sudah kami """

print("Sekian dan Teriman kasih")
```

Output :

```
PS D:\Kuliah\Tingkat 2\IF2124 - Teori Bahasa Formal dan Otomata\Tugas\
TUBES_TBFO> python parserprogram.py TestCase8.txt

Accepted
Time Execution:  0.043599699973128736 sec
```

Perlu diperhatikan, untuk comment dengan single line hanya bisa diterapkan bila pada ujung belakang perlu dituliskan \n. Selain itu perlu diberikan spasi pada tiap simbol yang melambangkan comment. Sebagai contoh :

- `'''<spasi><content><spasi>'''`
- `"""<spasi><content><spasi>"""`
- `#<spasi><content><spasi>\n`

4. Link Repository Github

Source code dapat diakses pada :
https://github.com/ghebyon/TUBES_TBFO

5. Pembagian Tugas

No	Nama	Tugas
1.	Gibran Darmawan	- Research Sumber(CFG2CNF convert) - Testing dan debug program - Pengerjaan CFG - Pengerjaan Laporan
2.	Ghebyon Tohada Nainggolan	- Pengerjaan Algoritma CYK - Membuat dan mengerjakan file “createToken.py” - Membuat dan mengerjakan file “main.py” - Pengerjaan Laporan - Testing dan debug program