Progress Report Tugas Besar Tahap 1 & 2

1. Kelompok:

No	NIM	Nama
1	1301180134	Abigael Mark Stevan
2	1301180072	Luqman Haries
3	1301180154	Muhammad Abdurrohman Al Fatih
4	1301180084	Prawiro Weninggalih

2. Spesifikasi Program

1.1 Parser Sederhana untuk Formula Logika Proposisi

Program Parser Sederhana untuk Formula Logika Proposisi adalah program yang menerapkan Finite Automata atau FA yang fungsinya untuk mengenali dan mengubah logika proposisi string berdasarkan Lexic dan Token Lexic-nya. Program ini akan melakukan parsing dan validasi terhadap Formula Logika Proposisi

Tujuan program tahap 1 adalah membangun program Lexical Analyzer dengan menerapkan Finite Automata untuk mengenali setiap Lexic yang dituliskan dalam formula. Hasil dari program tahap 1 ini adalah sequence Token Lexic berdasarkan string Formula yang dibaca.

Tujuan program tahap 2 ini adalah membuat program yang berfungsi sebagai Parser untuk melakukan validasi terhadap string Formula Logika Proposisi yang dimasukkan. Input program berupa sequence Token Lexic yang merupakan hasil keluaran dari Lexical Analyzer (program tahap 1). Untuk membangun sebuah Parser, program akan mengimplementasikan

Context Free Grammar atau Push Down Automata yang tepat sesuai aturan atau kebutuhan dalam mem-validasi sebuah string Formula Logika Proposisi.

1.2 Batasan Masalah

Batasan masalah dalam pembuatan program tahap 1 ini diantaranya :

1. Cukup menggunakan 1 buah FA yang dapat mengenali semua String Lexic berikut :

String Lexic	Jenis	Token	Keterangan
Proposisi p, q, r, s	operand	1	Hanya 1 simbol (di antara p, q, r, s) yang dikenal sebagai 1 proposisi
not	operator	2	Contoh penulisan yang diterima: o `not proposisi` o `not(proposisi)`
and	operator	3	Contoh penulisan yang diterima: o `proposisi and proposisi` o `(proposisi)and(proposisi)` o `(formula)and(formula)`
or	operator	4	Contoh penulisan yang diterima: o `proposisi or proposisi` o `(proposisi)or(proposisi)`

			o (formula)or(formula)
xor	operator	5	Contoh penulisan yang diterima: o proposisi xor proposisi o (proposisi)xor(proposisi) o (formula)xor(formula)
if	operator	6	Aturan penulisan yang diterima: o if proposisi then proposisi o if(proposisi)then(proposisi) o if(formula)then(formula)
then	operator	7	
iff	operator	8	Aturan penulisan yang diterima: o proposisi iff proposisi o (proposisi)iff(proposisi) o (formula)iff(formula)
(	grouping	9	Aturan penulisan yang diterima: o proposisi iff proposisi o (proposisi)iff(proposisi) o (formula)iff(formula)
)	grouping	10	

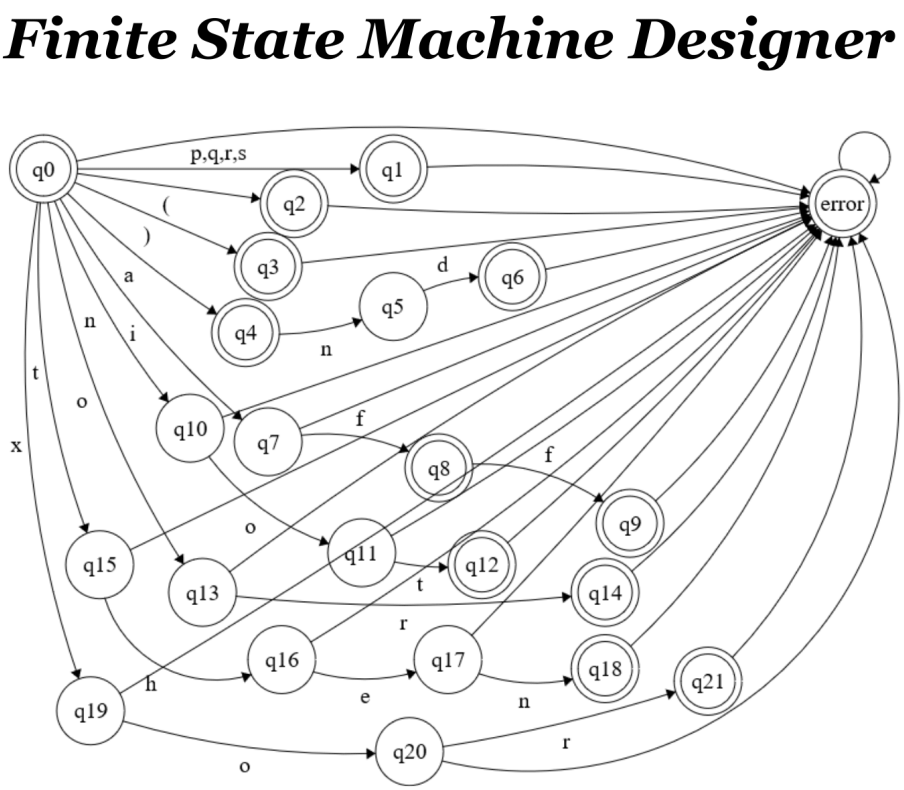
2. Tidak boleh menggunakan string matching
3. Program harus membaca satu per satu karakter/symbol dari string yang dibaca

1.3 Spesifikasi Kebutuhan Program

1. Python 3.X

3. Rancangan Finite Automata

1. Desain Finite Automata



2. Definisi Formal

	Definisi Formal
M =	(Q, ∑, S, F, δ)
Q =	{q0,q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11,q12,q13,q14,q15,q16,q17,q18,q19,q20,q21}
∑ =	{ pqrs, not, and, xor, or, if, then, iff, (,) }
S =	q0
F =	{q0,q1,q2,q3,q4,q6,q8,q9,q12,q14,q18,q21}
δ =	Tabel Transisi

3. Tabel transisi

Tabel Transisi

δ	p,q,r,s	n	o	t	a	n	d	o	r	x	o	r	i	f	t	h	e	n	i	f	f	(	)
{q0}	{q1}	{q10}	{q13}	{q15}	{q4}	{q10}	{}	{q13}	{}	{q19}	{q13}	{}	{q7}	{}	{q15}	{}	{}	{}	{q7}	{}	{}	{q2}	{q3}
{q1}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q2}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q3}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q4}	{}	{q5}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q5}	{}	{}	{}	{}	{}	{}	{q6}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q6}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q7}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{q8}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q8}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{q9}	{}	{}
{q9}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q10}	{}	{}	{q11}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q11}	{}	{}	{}	{q12}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q12}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q13}	{}	{}	{}	{}	{}	{}	{}	{}	{q14}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q14}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q15}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{q16}	{}	{}	{}	{}	{}	{}	{}
{q16}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{q17}	{}	{}	{}	{}	{}	{}
{q17}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{q18}	{}	{}	{}	{}	{}
{q18}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q19}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{q20}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q20}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{q21}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}
{q21}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}	{}

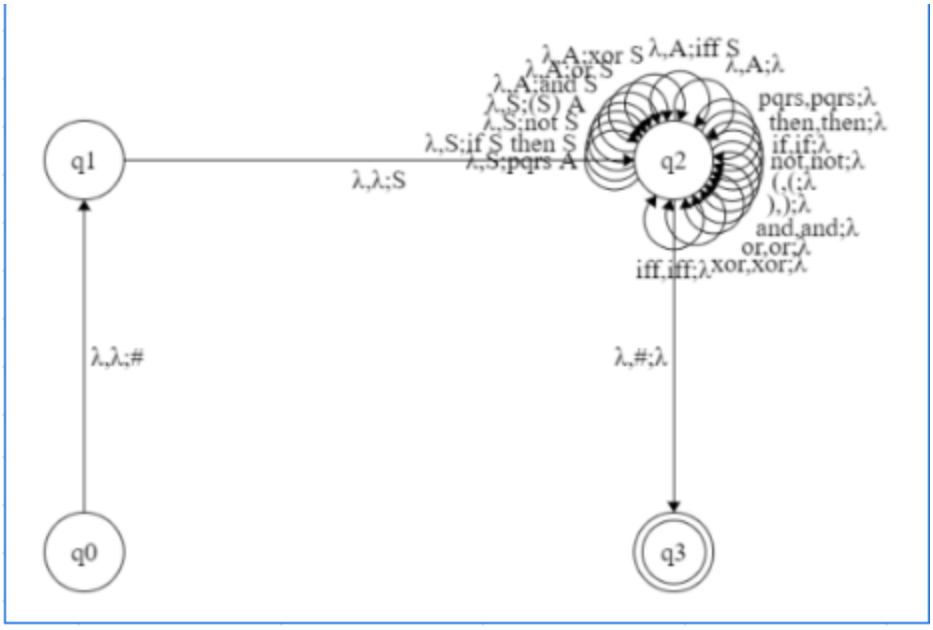
4. Rancangan Context Free Grammar dan Pushdown Automata

1. Definisi Formal Context Free Grammar

   S -> operand A | if S then S | not S | (S) A

   A -> and S | or S | xor S | iff S | Kosong

2. Desain Pushdown Automata



3. Definisi Formal Pushdown Automata

Definisi Formal untuk PDA		
M =	(S,∑, Γ,i,F,T)	
S =	{q0,q1,q2,q3}	//state
∑ =	{pqrs, if, then, not, (,), and, or, xor, iff }	//simbol terminal
Γ=	{λ,#,S,A}	//simbol stack
i =	{q0}	//State awal
F =	{q3}	//Final State
T =		//Transisi

Transisi	
(q0,λ,λ;q1,#)	(q2,prqs,prqs;q2,λ)
(q1,λ,λ;q2,S)	(q2,if,if;q2,λ)
(q2,λ,S;q2, pqrs A)	(q2,then,then;q2,λ)
(q2,λ,S;q2, if S then S)	(q2,not,not;q2,λ)
(q2,λ,S;q2, not S)	(q2,(,(;q2, λ)
(q2,λ,S;q2, (S) A)	(q2,),);q2,λ)
(q2,λ,A;q2, and S)	(q2,and,and;q2,λ)
(q2,λ,A;q2, or S)	(q2,or,or;q2,λ)
(q2,λ,A;q2, xor S)	(q2,xor,xor;q2,λ)
(q2,λ,A;q2, iff S)	(q2,iff,iff;q2,λ)
(q2,λ,A;q2, λ)	(q2,λ,#;q3, λ)
	(dari state, baca, pop ; ke state, push)

5. Cara Kerja Program

Program bekerja dengan menerima input serangkaian string dan memisahkan string-string tersebut berdasarkan spasi (*exclusive*), kurung buka (*inclusive*), dan kurung tutup(inclusive) lalu memasukkannya ke dalam list of string. Cara memisahkan stringnya adalah dengan membaca 1 per 1 karakter dan apabila bertemu dengan spasi, kurung buka, atau kurung tutup maka karakter-karakter sebelumnya bisa dianggap sebagai satu buah string dan dimasukkan ke dalam list of string. Setelah memisah-misahkan string, tiap-tiap string akan dianalisa apakah string tersebut merupakan *lexic* atau bukan, jika merupakan lexic maka program akan mengeluarkan token lexicnya, jika tidak maka program akan berhenti menganalisa list of string tersebut. Untuk tahap validasinya, menerapkan konsep PDA dengan production rule dari CFG yang sudah kami buat

6. Pengujian Program

Input	**Output Program**	**Keterangan**

p and q or r	1, 3, 1, 4, 1 Valid	Sesuai, hasil pemecahan string tersebut adalah ['p', 'and', 'q', 'or', 'r'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
if p then (not q s)	6, 1, 7, 9, 2, 1, 1, 10 Tidak Valid	Sesuai, hasil pemecahan string tersebut adalah ['if', 'p', 'then', '(', 'not', 'q', 's', ')'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
p xor (q and not(p and q))	1, 5, 9, 1, 3, 2, 9, 1, 3, 1, 10, 10 Valid	Sesuai, hasil pemecahan string tersebut adalah ['p', 'xor', '(', 'q', 'and', 'not', '(', 'p', 'and', 'q', ')', ')'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
(p and q ifg(r or s)	9, 1, 3, 1, -1 Tidak Valid	Sesuai, hasil pemecahan string tersebut adalah ['(', 'p', 'and', 'q', 'ifg', '(', 'r', 'or', 's', ')']

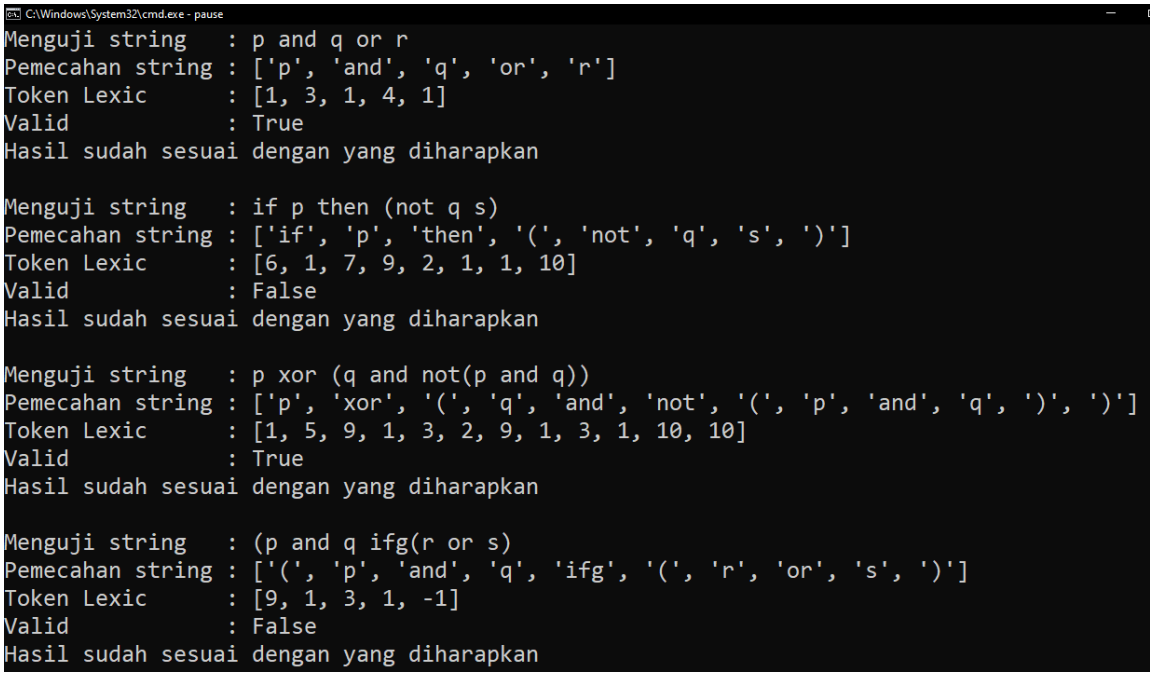
		Dan menghasilkan token yang sesuai, validasinya juga sesuai
not iff p	2, 8, 1 Tidak Valid	Sesuai, hasil pemecahan string tersebut adalah ['not', 'iff', 'p'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
not not not q xor not p	2, 2, 2, 1, 5, 2, 1 Valid	Sesuai, hasil pemecahan string tersebut adalah ['not', 'not', 'not', 'q', 'xor', 'not', 'p'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
if p then r	6, 1, 7, 1 Valid	Sesuai, hasil pemecahan string tersebut adalah ['if', 'p', 'then', 'r'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
if (if r then s) and p then q	6, 9, 6, 1, 7, 1, 10, 3, 1, 7, 1 Valid	Sesuai, hasil pemecahan string tersebut adalah ['if', '(', 'if', 'r', 'then', 's', ')', 'and', 'p', 'then', 'q'] Dan menghasilkan token yang sesuai, validasinya juga sesuai

p and q iff q	1, 3, 1, 8, 1 Valid	Sesuai, hasil pemecahan string tersebut adalah ['p', 'and', 'q', 'iff', 'q'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
if (if p then r) then r	6, 9, 6, 1, 7, 1, 10, 7, 1 Valid	Sesuai, hasil pemecahan string tersebut adalah ['if', '(', 'if', 'p', 'then', 'r', ')', 'then', 'r'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
(p and q) and q	9, 1, 3, 1, 10, 3, 1 Valid	Sesuai, hasil pemecahan string tersebut adalah ['(', 'p', 'and', 'q', ')', 'and', 'q'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
(((p and q) or r) and s)	9, 9, 9, 1, 3, 1, 10, 4, 1, 10, 3, 1, 10 Valid	Sesuai, hasil pemecahan string tersebut adalah ['(', '(', '(', 'p', 'and', 'q', ')', 'or', 'r', ')', 'and', 's', ')'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
(p and q) iff q	9, 1, 3, 1, 10, 8, 1	Sesuai, hasil pemecahan string tersebut adalah

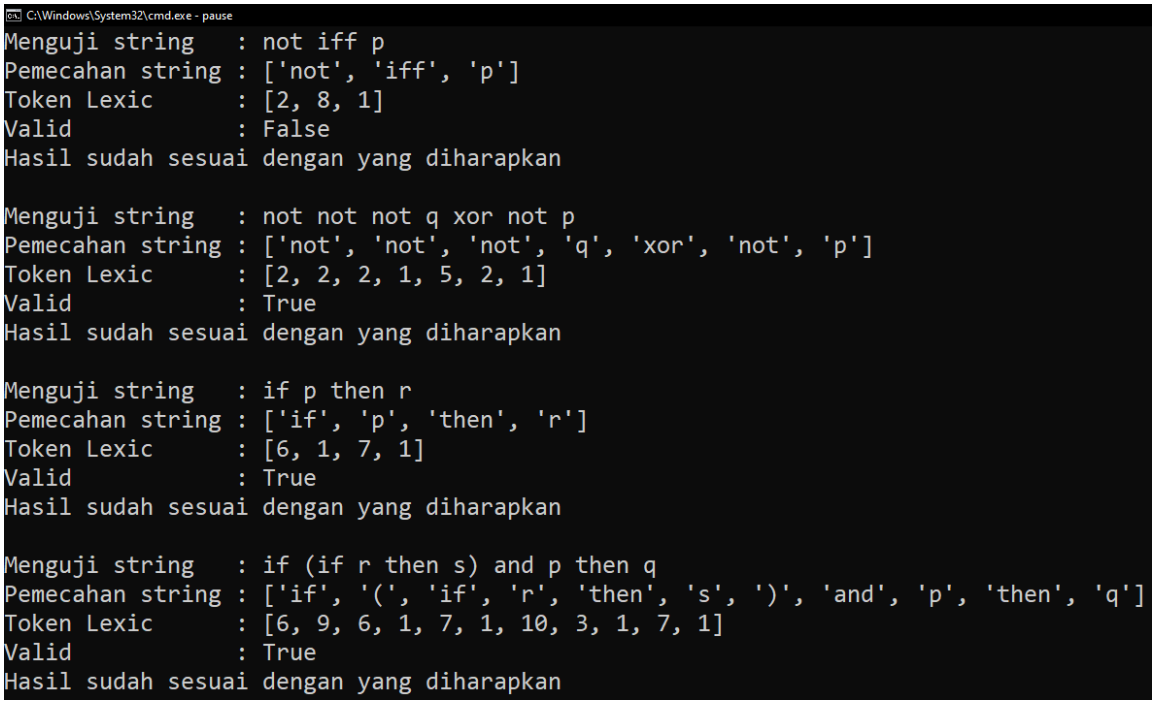
	Valid	['(', 'p', 'and', 'q', ')', 'iff', 'q'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
p and if p iff q then r	1, 3, 6, 1, 8, 1, 7, 1 Valid	Sesuai, hasil pemecahan string tersebut adalah ['p', 'and', 'if', 'p', 'iff', 'q', 'then', 'r'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
not (p)	2, 9, 1, 10 Valid	Sesuai, hasil pemecahan string tersebut adalah ['not', '(', 'p', ')'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
not p	2, 1 Valid	Sesuai, hasil pemecahan string tersebut adalah ['not', 'p'] Dan menghasilkan token yang sesuai, validasinya juga sesuai
not (if p then q)	2, 9, 6, 1, 7, 1, 10 Valid	Sesuai, hasil pemecahan string tersebut adalah ['not', '(', 'if', 'p', 'then', 'q', ')']

		Dan menghasilkan token yang sesuai, validasinya juga sesuai

7. Screenshot Program



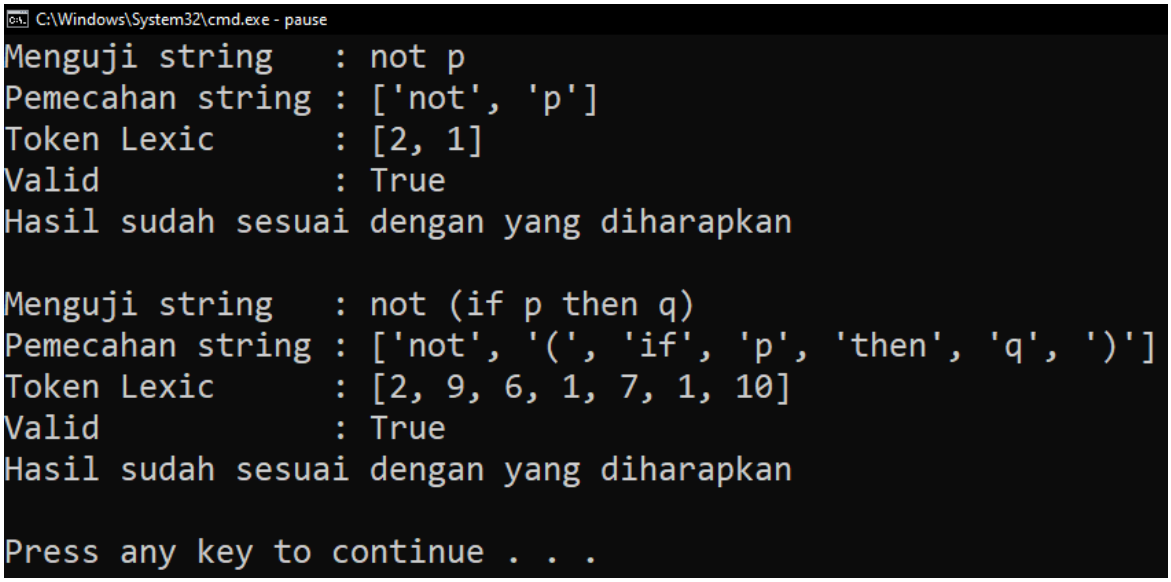
```
C:\Windows\System32\cmd.exe - pause
Menguji string   : p and q or r
Pemecahan string : ['p', 'and', 'q', 'or', 'r']
Token Lexic      : [1, 3, 1, 4, 1]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan

Menguji string   : if p then (not q s)
Pemecahan string : ['if', 'p', 'then', '(', 'not', 'q', 's', ')']
Token Lexic      : [6, 1, 7, 9, 2, 1, 1, 10]
Valid            : False
Hasil sudah sesuai dengan yang diharapkan

Menguji string   : p xor (q and not(p and q))
Pemecahan string : ['p', 'xor', '(', 'q', 'and', 'not', '(', 'p', 'and', 'q', ')', ')']
Token Lexic      : [1, 5, 9, 1, 3, 2, 9, 1, 3, 1, 10, 10]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan

Menguji string   : (p and q ifg(r or s)
Pemecahan string : ['(', 'p', 'and', 'q', 'ifg', '(', 'r', 'or', 's', ')']
Token Lexic      : [9, 1, 3, 1, -1]
Valid            : False
Hasil sudah sesuai dengan yang diharapkan
```



```
C:\Windows\System32\cmd.exe - pause
Menguji string   : not iff p
Pemecahan string : ['not', 'iff', 'p']
Token Lexic      : [2, 8, 1]
Valid            : False
Hasil sudah sesuai dengan yang diharapkan

Menguji string   : not not not q xor not p
Pemecahan string : ['not', 'not', 'not', 'q', 'xor', 'not', 'p']
Token Lexic      : [2, 2, 2, 1, 5, 2, 1]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan

Menguji string   : if p then r
Pemecahan string : ['if', 'p', 'then', 'r']
Token Lexic      : [6, 1, 7, 1]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan

Menguji string   : if (if r then s) and p then q
Pemecahan string : ['if', '(', 'if', 'r', 'then', 's', ')', 'and', 'p', 'then', 'q']
Token Lexic      : [6, 9, 6, 1, 7, 1, 10, 3, 1, 7, 1]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan
```

```
C:\Windows\System32\cmd.exe - pause
Menguji string   : p and q iff q
Pemecahan string : ['p', 'and', 'q', 'iff', 'q']
Token Lexic      : [1, 3, 1, 8, 1]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan

Menguji string   : if (if p then r) then r
Pemecahan string : ['if', '(', 'if', 'p', 'then', 'r', ')', 'then', 'r']
Token Lexic      : [6, 9, 6, 1, 7, 1, 10, 7, 1]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan

Menguji string   : p and if p iff q then r
Pemecahan string : ['p', 'and', 'if', 'p', 'iff', 'q', 'then', 'r']
Token Lexic      : [1, 3, 6, 1, 8, 1, 7, 1]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan

Menguji string   : (p and q) and q
Pemecahan string : ['(', 'p', 'and', 'q', ')', 'and', 'q']
Token Lexic      : [9, 1, 3, 1, 10, 3, 1]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan
```

```
C:\Windows\System32\cmd.exe - pause
Menguji string   : (((p and q) or r) and s)
Pemecahan string : ['(', '(', '(', 'p', 'and', 'q', ')', 'or', 'r', ')', 'and', 's', ')']
Token Lexic      : [9, 9, 9, 1, 3, 1, 10, 4, 1, 10, 3, 1, 10]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan

Menguji string   : (p and q) iff q
Pemecahan string : ['(', 'p', 'and', 'q', ')', 'iff', 'q']
Token Lexic      : [9, 1, 3, 1, 10, 8, 1]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan

Menguji string   : p and if p iff q then r
Pemecahan string : ['p', 'and', 'if', 'p', 'iff', 'q', 'then', 'r']
Token Lexic      : [1, 3, 6, 1, 8, 1, 7, 1]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan

Menguji string   : not (p)
Pemecahan string : ['not', '(', 'p', ')']
Token Lexic      : [2, 9, 1, 10]
Valid            : True
Hasil sudah sesuai dengan yang diharapkan
```



8. Kodingan Program

Kodingan program dapat dilihat di :

https://drive.google.com/file/d/1JHVM3mYdQ1R2EKX5PCvf3sWwWqG_cvwB/view?usp=sharing