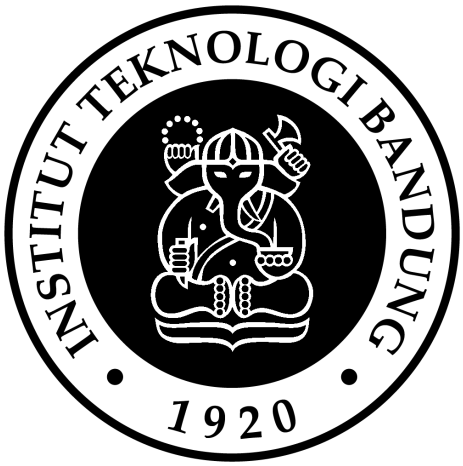
JAVASCRIPT PARSER

LAPORAN TUGAS BESAR

Diajukan sebagai salah satu tugas mata kuliah Teori Bahasa Formal dan Otomata pada Semester II Tahun Akademik 2022/2023

oleh

Henry Anand Septian Radityo	**13521004**
Matthew Mahendra	**13521007**
Ahmad Nadil	**13521024**



SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG

2022

DAFTAR ISI

BAB I
TEORI DASAR

I.I Finite Automata

Finite automata (FA) merupakan sebuah mesin yang digunakan untuk mendeteksi dan melakukan validasi terhadap sebuah pola. Sebuah FA memiliki aturan yang digunakan untuk berpindah antar *state*. Pola yang diterima oleh mesin diarahkan ke sebuah *state* yaitu *final state*. Aturan perpindahan ini disimbolkan sebagai δ dan setiap state disimbolkan q dengan himpunan dari state disimbolkan sebagai Q. Dari pengertian di atas, FA dapat dinyatakan secara formal menggunakan *tuple* dengan lima elemen sebagai berikut,

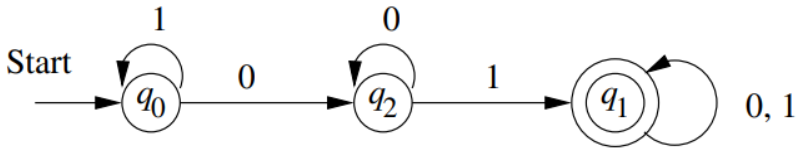
$$A = (Q, \Sigma, \delta, q_0, q_f)$$

dengan Q, δ sama dengan pengertian di atas, Σ sebagai simbol-simbol yang dapat di-input, q_0 sebagai input *state* (*state* yang menerima masukan), q_f sebagai *final state*.

Representasi FA dapat dinyatakan melalui tabel transisi ataupun diagram transisi. Dengan menggunakan tabel transisi, setiap simbol masukan dan *state* dipasangkan sesuai dengan δ. Dengan menggunakan diagram transisi, dibuat diagram sesuai dengan δ.

	0	1
→ q_0	q_2	q_0
*q_1	q_1	q_1
q_2	q_2	q_1

Gambar 1. Contoh Tabel Transisi FA (Hopcroft, et. al., 2007)



Gambar 2. Contoh Diagram Transisi FA (Hopcroft, et. al., 2007)

I.II Context Free Grammar

Context Free Grammar (CFG) merupakan *grammar* yang digunakan untuk mendefinisikan *context free language* (CFL) secara formal. CFG memiliki struktur rekursi dalam mendefinisikan CFL. Struktur dari CFG meliputi beberapa hal yaitu, simbol terminal T yang digunakan untuk membangun alfabet dalam bahasa CFL, variabel V yang digunakan merepresentasikan kata dalam bahasa CFL, simbol mulai S, dan production P yang digunakan untuk menggantikan setiap variabel untuk menghasilkan suatu bahasa.

Untuk setiap CFG yang tidak menghasilkan ε, maka CFG dapat dibentuk menjadi Chomsky Normal Form (CNF). Untuk CNF, setiap aturan hanya boleh mengandung maksimal 2 variabel dan untuk jumlah variabel yang lebih dari 2, maka variabel tersebut dihubungkan ke sebuah aturan baru yang dapat menghasilkan variabel yang lebih tersebut.

Pengujian apakah suatu *sentence* termasuk dalam CFL dapat memanfaatkan CNF. CNF yang telah dibuat dijalankan pada algoritma Cocke-Younger-Kasami (CYK). CYK ini membentuk suatu tabel berdasarkan CNF. Apabila suatu *sentence* terdapat dalam CFL, maka S akan terdapat pada ujung kiri atas pada tabel CYK.

$\{S,A,C\}$				
–	$\{S,A,C\}$			
–	$\{B\}$	$\{B\}$		
$\{S,A\}$	$\{B\}$	$\{S,C\}$	$\{S,A\}$	
$\{B\}$	$\{A,C\}$	$\{A,C\}$	$\{B\}$	$\{A,C\}$
b	*a*	*a*	*b*	*a*

Gambar 3. Contoh tabel CYK untuk suatu CFL (Hopcroft, et. al., 2007)

I.III Syntax Javascript

Bahasa pemrograman Javascript dapat dianalisa kebenaran *syntax*-nya menggunakan CFG dan FA. Bahasa Javascript menggunakan bentuk *block* yang ditandai oleh tanda kurung kurawal '{}'

untuk beberapa sintaksnya. Maka dari itu, perlu dilakukan handling terkait hal tersebut dalam pembuatan parser agar setiap tanda kurung kurawal selalu berpasangan. Selain itu, end of line (EOL) pada Javascript dapat dinyatakan menggunakan *semi-colon* atau baris baru. Kelompok mengimplementasikan *semi-colon* sebagai EOL program. Nama variabel harus dimulai dengan huruf atau *underscore* dan tidak boleh dimulai dengan angka.

Berikut akan diberikan sintaks-sintaks Javascript umum beserta perhatian khusus pada pembuatan grammarnya,

1. Conditional

```
if (x==2){
    x+=1;
}else if(x == 3){
    x+=2;
}else{
    x+=3;
```

Untuk conditional, perlu dilakukan handling untuk conditional yang hanya memiliki ‘if’ tunggal, ‘if dan else if’, ‘if dan else’, dan ‘if, else if, dan else’. Perlu diperhatikan adanya kurung kurawal dan kemungkinan kalimat logika pada conditional, maupun kalimat prosedural pada body if

2. For loop

```
for (i = 0; i < 5; i++){
    x += 1;
    break;
}
```

Untuk for loop, sintaks ekspresi perulangan dipisahkan oleh *semi-colon* dan menggunakan *block* sebagai bodynya. Body dari for loop memungkinkan untuk adanya perintah break.

3. While dan do-while

```
while (booleanX){
    console.log(“Hai”);
    booleanX = evaluate();
}
```

```
do{
     x+=1;
} while (x < 1);
```

Untuk while dan do-while, perlu diperhatikan kondisi perulangannya serta sintaks kurung dan sintaks ‘do’ dan atau ‘while’ pada setiap pemanggilan perulangan ini. Body dari while dan do-while memungkinkan adanya break.

4. Deklarasi function

```
function MyFunction(param){
    return false;
}
```

Untuk nama function, harus dilakukan validasi agar sesuai kaidah (kaidah sama dengan nama variabel). Dimungkinkan untuk melakukan return dalam suatu function.

5. Switch

```
switch (x):
    case 1:
        console.log(“Hidup”);
        break;
    case 2:
        console.log(“Informatika”);
        break;
    default:
        console.log(“ITB Almamater Tercinta”);
```

Pada switch, ekspresi test dapat berupa variabel atau hasil ekspresi. Ekspresi case dapat berupa integer, string, boolean ataupun object. Dimungkinkan untuk melakukan break di dalam switch.

6. Deklarasi Kelas

```
class Rectangle {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }
        // Getter
    get area() {
        return this.calcArea();
```

```
    }
        // Method
    calcArea() {
        return this.height * this.width;
    }
}

let Rectangle = class {
    constructor(height, width) {
      this.height = height;
      this.width = width;
    }
};
console.log(Rectangle.name); // "Rectangle"


  // named
Rectangle = class Rectangle2 {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }
};
```

Pada kelas, perlu diperhatikan macam method yang dapat dituliskan dalam body kelas dan juga macam cara deklarasi kelas, menggunakan assignment maupun langsung menamakannya.

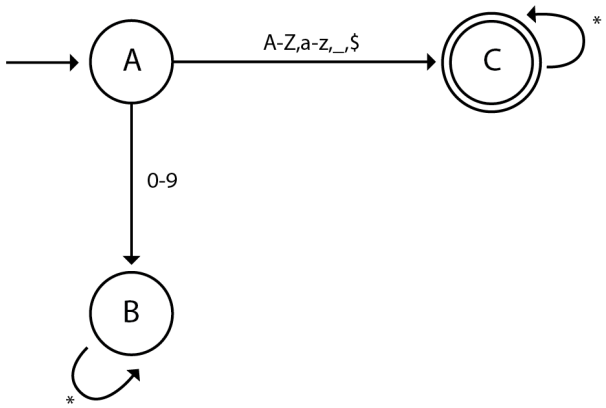
7. Integer

   Dalam Javascript, integer dapat dimulai dengan angka 0, misalnya untuk 09, akan tetap dibaca 9. Begitu pula untuk 0000000010, akan dibaca sebagai 10.

BAB II
IMPLEMENTASI FA, CFG, DAN PROGRAM

II.I Implementasi FA

FA digunakan untuk melakukan validasi terhadap nama variabel dengan aturan yang sudah dijelaskan pada upabab I.III mengenai Syntax Javascript. Berikut adalah implementasi FA pada tugas ini,



Gambar 4. Implementasi DFA untuk penamaan variabel

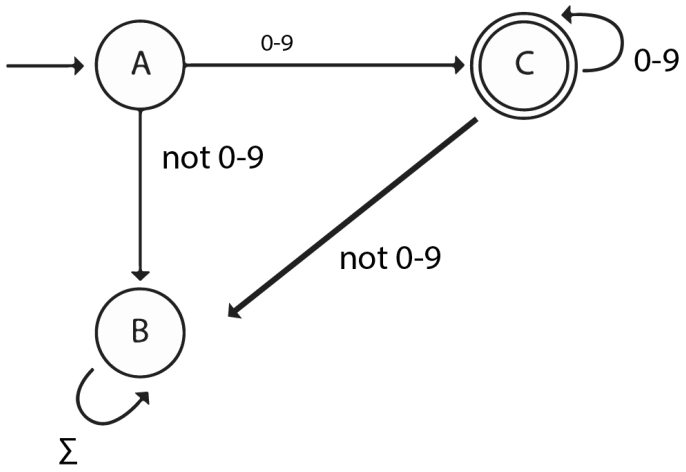
Pada DFA dapat dilihat *start state* berada di state A dan ketika nama variabel diawali dengan angka, maka akan ditolak menuju ke *death state*. Apabila penamaan diawali dengan alfabet, underscore, dan juga tanda dollar maka akan memasuki *final state*. Tanda *asterisk* (*) digunakan untuk menyimbolkan semua input diantaranya alfabet, *underscore*, dollar, dan juga angka dikarenakan aturan penamaan variabel dari Javascript mengharuskan pengguna untuk memberi penamaan dengan beberapa karakter tersebut.

Pada implementasi untuk kode program, kami mengubah struktur dari DFA tersebut menjadi sebuah regular ekspresi. Apabila aturan dari DFA tersebut dibuat dalam regular ekspresi akan menghasilkan seperti berikut,

```
[A-Za-z_$][A-Za-z0-9_$]*
```

Kemudian, regular ekspresi tersebut diimplementasikan pada lexer untuk mendeteksi adanya penamaan variabel yang valid dan tidak.

Selain itu, FA digunakan pula untuk menentukan apakah suatu input merupakan string atau bukan. Ciri-ciri integer adalah hanya memiliki angka saja. Oleh sebab itu, dapat dibuat diagram transisi sebagai berikut,



Gambar 5. Diagram Transisi DFA untuk Integer

dari FA yang dibuat, dibuat tiga regular expression untuk melakukan validasi sebagai berikut

```
[0-9]*\.[0-9]+
[0-9][0-9]+
[0-9]
```

II.II Implementasi CFG

CFG digunakan untuk melakukan pembangunan syntax yang sesuai dengan Javascript. Berikut adalah implementasi dari CFG pada tugas ini,

```
S -> S ENTER
S -> ENTER S
S -> ENTER
S -> ST_PROG
ST_PROG -> STATEMENT_BLOCK_IF
ST_PROG -> STATEMENT_FOR
ST_PROG -> STATEMENT_WHILE
ST_PROG -> STATEMENT_FUNC
ST_PROG -> STATEMENT_SWITCH
ST_PROG -> STATEMENT_INVOKE_FUNC
ST_PROG -> SENTENCE
```

```
STATEMENT_RETURN -> RETURN RETURN_VAR TITIK_KOMA
STATEMENT_RETURN -> RETURN SENTENCE
STATEMENT_RETURN -> RETURN KURUNG_BUKA RETURN_VAR KURUNG_TUTUP TITIK_KOMA
STATEMENT_IF -> IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP KURUNG_KURAWAL_BUKA
IF_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_IF -> IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP KURUNG_KURAWAL_BUKA
ENTER IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_IF -> IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP KURUNG_KURAWAL_BUKA
ENTER IF_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_IF -> IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP KURUNG_KURAWAL_BUKA
IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_IF -> IF KURUNG_BUKA LOGIC_OPERATOR KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_IF -> IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_IF -> IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_IF -> IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA IF_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP ENTER
STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP ENTER
STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE KURUNG_KURAWAL_TUTUP ENTER
STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE KURUNG_KURAWAL_TUTUP STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
```

```
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP ENTER
STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP ENTER
STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE KURUNG_KURAWAL_TUTUP ENTER
STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE KURUNG_KURAWAL_TUTUP STATEMENT_ELSE_IF
STATEMENT_ELSE_IF -> ELSE IF KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE_IF
STATEMENT_ELSE -> ELSE KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE -> ELSE KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE
KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE -> ELSE KURUNG_KURAWAL_BUKA IF_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE -> ELSE ENTER KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE -> ELSE ENTER KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE
KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE -> ELSE ENTER KURUNG_KURAWAL_BUKA IF_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE -> ENTER ELSE ENTER KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE -> ENTER ELSE ENTER KURUNG_KURAWAL_BUKA ENTER IF_SENTENCE
KURUNG_KURAWAL_TUTUP
STATEMENT_ELSE -> ENTER ELSE ENTER KURUNG_KURAWAL_BUKA IF_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_INVOKE_FUNC -> FUNC_NAME TITIK_KOMA
STATEMENT_THROW -> THROW THROW_VAR TITIK_KOMA
STATEMENT_BLOCK_IF -> STATEMENT_IF
STATEMENT_BLOCK_IF -> STATEMENT_IF STATEMENT_ELSE
STATEMENT_BLOCK_IF -> STATEMENT_IF STATEMENT_ELSE_IF STATEMENT_ELSE
STATEMENT_BLOCK_IF -> STATEMENT_IF STATEMENT_ELSE_IF
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
INC KURUNG_TUTUP KURUNG_KURAWAL_BUKA ENTER FOR_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
INC KURUNG_TUTUP KURUNG_KURAWAL_BUKA FOR_SENTENCE KURUNG_KURAWAL_TUTUP
```

```
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
INC KURUNG_TUTUP KURUNG_KURAWAL_BUKA ENTER FOR_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
INC KURUNG_TUTUP KURUNG_KURAWAL_BUKA FOR_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
DEC KURUNG_TUTUP KURUNG_KURAWAL_BUKA ENTER FOR_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
DEC KURUNG_TUTUP KURUNG_KURAWAL_BUKA FOR_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
DEC KURUNG_TUTUP KURUNG_KURAWAL_BUKA ENTER FOR_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
DEC KURUNG_TUTUP KURUNG_KURAWAL_BUKA FOR_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
INC KURUNG_TUTUP ENTER KURUNG_KURAWAL_BUKA ENTER FOR_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
INC KURUNG_TUTUP ENTER KURUNG_KURAWAL_BUKA FOR_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
INC KURUNG_TUTUP ENTER KURUNG_KURAWAL_BUKA ENTER FOR_SENTENCE
KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
INC KURUNG_TUTUP ENTER KURUNG_KURAWAL_BUKA FOR_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
DEC KURUNG_TUTUP ENTER KURUNG_KURAWAL_BUKA ENTER FOR_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
DEC KURUNG_TUTUP ENTER KURUNG_KURAWAL_BUKA FOR_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
DEC KURUNG_TUTUP ENTER KURUNG_KURAWAL_BUKA ENTER FOR_SENTENCE
KURUNG_KURAWAL_TUTUP
STATEMENT_FOR -> FOR KURUNG_BUKA ASSIGNMENT_SENTENCE LOGIC_SENTENCE TITIK_KOMA
DEC KURUNG_TUTUP ENTER KURUNG_KURAWAL_BUKA FOR_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_FUNC -> FUNC FUNC_NAME KURUNG_KURAWAL_BUKA ENTER FUNC_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_FUNC -> FUNC FUNC_NAME KURUNG_KURAWAL_BUKA FUNC_SENTENCE
KURUNG_KURAWAL_TUTUP
STATEMENT_FUNC -> FUNC FUNC_NAME KURUNG_KURAWAL_BUKA ENTER FUNC_SENTENCE
KURUNG_KURAWAL_TUTUP
STATEMENT_FUNC -> FUNC FUNC_NAME KURUNG_KURAWAL_BUKA FUNC_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_FUNC -> FUNC FUNC_NAME ENTER KURUNG_KURAWAL_BUKA ENTER FUNC_SENTENCE
ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_FUNC -> FUNC FUNC_NAME ENTER KURUNG_KURAWAL_BUKA FUNC_SENTENCE
KURUNG_KURAWAL_TUTUP
STATEMENT_FUNC -> FUNC FUNC_NAME ENTER KURUNG_KURAWAL_BUKA ENTER FUNC_SENTENCE
KURUNG_KURAWAL_TUTUP
STATEMENT_FUNC -> FUNC FUNC_NAME ENTER KURUNG_KURAWAL_BUKA FUNC_SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_WHILE -> WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA ENTER WHILE_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_WHILE -> WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
```

```
KURUNG_KURAWAL_BUKA WHILE_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_WHILE -> WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA ENTER WHILE_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_WHILE -> WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
KURUNG_KURAWAL_BUKA WHILE_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_WHILE -> DO KURUNG_KURAWAL_BUKA ENTER WHILE_SENTENCE
KURUNG_KURAWAL_TUTUP ENTER WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
TITIK_KOMA
STATEMENT_WHILE -> DO KURUNG_KURAWAL_BUKA WHILE_SENTENCE KURUNG_KURAWAL_TUTUP
WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP TITIK_KOMA
STATEMENT_WHILE -> DO KURUNG_KURAWAL_BUKA WHILE_SENTENCE KURUNG_KURAWAL_TUTUP
ENTER WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP TITIK_KOMA
STATEMENT_WHILE -> DO KURUNG_KURAWAL_BUKA ENTER WHILE_SENTENCE
KURUNG_KURAWAL_TUTUP WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP TITIK_KOMA
STATEMENT_WHILE -> WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA ENTER WHILE_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_WHILE -> WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA WHILE_SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_WHILE -> WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA ENTER WHILE_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_WHILE -> WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP ENTER
KURUNG_KURAWAL_BUKA WHILE_SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_WHILE -> DO ENTER KURUNG_KURAWAL_BUKA ENTER WHILE_SENTENCE
KURUNG_KURAWAL_TUTUP ENTER WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
TITIK_KOMA
STATEMENT_WHILE -> DO ENTER KURUNG_KURAWAL_BUKA WHILE_SENTENCE
KURUNG_KURAWAL_TUTUP WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP TITIK_KOMA
STATEMENT_WHILE -> DO ENTER KURUNG_KURAWAL_BUKA WHILE_SENTENCE
KURUNG_KURAWAL_TUTUP ENTER WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
TITIK_KOMA
STATEMENT_WHILE -> DO ENTER KURUNG_KURAWAL_BUKA ENTER WHILE_SENTENCE
KURUNG_KURAWAL_TUTUP WHILE KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP TITIK_KOMA
DECLARE_ARRAY -> ARRAY_VAR
DECLARE_ARRAY -> ARRAY_VAR KOMA DECLARE_ARRAY
DECLARE_ARRAY -> OPEN_BRACKET DECLARE_ARRAY KOMA CLOSE_BRACKET
DECLARE_ARRAY -> OPEN_BRACKET DECLARE_ARRAY CLOSE_BRACKET
STATEMENT_ARRAY -> OPEN_BRACKET DECLARE_ARRAY CLOSE_BRACKET TITIK_KOMA
STATEMENT_ARRAY -> OPEN_BRACKET DECLARE_ARRAY CLOSE_BRACKET
STATEMENT_ARRAY -> OPEN_BRACKET CLOSE_BRACKET TITIK_KOMA
STATEMENT_CASE -> CASE CASE_EXPR TITIK_DUA ENTER SENTENCE ENTER BREAK
TITIK_KOMA
STATEMENT_CASE -> CASE CASE_EXPR TITIK_DUA ENTER SENTENCE
STATEMENT_CASE -> CASE CASE_EXPR TITIK_DUA ENTER SENTENCE ENTER BREAK
TITIK_KOMA ENTER STATEMENT_CASE
STATEMENT_CASE -> CASE CASE_EXPR TITIK_DUA ENTER SENTENCE ENTER BREAK
TITIK_KOMA ENTER STATEMENT_DEFAULT
STATEMENT_DEFAULT -> DEFAULT TITIK_DUA ENTER SENTENCE
STATEMENT_DEFAULT -> DEFAULT TITIK_DUA ENTER SENTENCE ENTER BREAK TITIK_KOMA
ENTER STATEMENT_CASE
STATEMENT_DEFAULT -> DEFAULT TITIK_DUA ENTER SENTENCE ENTER TITIK_KOMA ENTER
STATEMENT_CASE
STATEMENT_DEFAULT -> DEFAULT TITIK_DUA ENTER SENTENCE ENTER BREAK TITIK_KOMA
STATEMENT_SWITCH -> SWITCH KURUNG_BUKA SWITCH_EXPR KURUNG_TUTUP
KURUNG_KURAWAL_BUKA ENTER STATEMENT_CASE ENTER KURUNG_KURAWAL_TUTUP
```

```
STATEMENT_SWITCH -> SWITCH KURUNG_BUKA SWITCH_EXPR KURUNG_TUTUP
KURUNG_KURAWAL_BUKA STATEMENT_CASE KURUNG_KURAWAL_TUTUP
STATEMENT_SWITCH -> SWITCH KURUNG_BUKA SWITCH_EXPR KURUNG_TUTUP
KURUNG_KURAWAL_BUKA ENTER STATEMENT_CASE KURUNG_KURAWAL_TUTUP
STATEMENT_SWITCH -> SWITCH KURUNG_BUKA SWITCH_EXPR KURUNG_TUTUP
KURUNG_KURAWAL_BUKA STATEMENT_CASE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_TRY -> TRY KURUNG_KURAWAL_BUKA ENTER SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_TRY -> TRY KURUNG_KURAWAL_BUKA ENTER SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_TRY -> TRY KURUNG_KURAWAL_BUKA SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_TRY -> TRY KURUNG_KURAWAL_BUKA SENTENCE KURUNG_BUKA
STATEMENT_CATCH -> CATCH KURUNG_BUKA VAR KURUNG_TUTUP KURUNG_KURAWAL_BUKA
ENTER SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_CATCH -> CATCH KURUNG_BUKA VAR KURUNG_TUTUP KURUNG_KURAWAL_BUKA
ENTER SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_CATCH -> CATCH KURUNG_BUKA VAR KURUNG_TUTUP KURUNG_KURAWAL_BUKA
SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_CATCH -> CATCH KURUNG_BUKA VAR KURUNG_TUTUP KURUNG_KURAWAL_BUKA
ENTER SENTENCE ENTER KURUNG_KURAWAL_TUTUP
STATEMENT_FINALLY -> FINALLY KURUNG_KURAWAL_BUKA SENTENCE KURUNG_KURAWAL_TUTUP
STATEMENT_FINALLY -> FINALLY KURUNG_KURAWAL_BUKA ENTER SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_FINALLY -> FINALLY KURUNG_KURAWAL_BUKA ENTER SENTENCE
KURUNG_KURAWAL_TUTUP
STATEMENT_FINALLY -> FINALLY KURUNG_KURAWAL_BUKA SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_TRY_CATCH -> STATEMENT_TRY STATEMENT_CATCH
STATEMENT_TRY_CATCH -> STATEMENT_TRY ENTER STATEMENT_CATCH
STATEMENT_TRY_FINALLY -> STATEMENT_TRY STATEMENT_FINALLY
STATEMENT_TRY_FINALLY -> STATEMENT_TRY ENTER STATEMENT_FINALLY
STATEMENT_TRY_CATCH_FINALLY -> STATEMENT_TRY STATEMENT_CATCH STATEMENT_FINALLY
STATEMENT_TRY_CATCH_FINALLY -> STATEMENT_TRY ENTER STATEMENT_CATCH ENTER
STATEMENT_FINALLY
STATEMENT_TRY_CATCH_FINALLY -> STATEMENT_TRY STATEMENT_CATCH ENTER
STATEMENT_FINALLY
STATEMENT_TRY_CATCH_FINALLY -> STATEMENT_TRY ENTER STATEMENT_CATCH
STATEMENT_FINALLY
STATEMENT_CLASS -> CLASS VAR KURUNG_KURAWAL_BUKA ENTER STATEMENT_METHODS ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_CLASS -> CLASS VAR KURUNG_KURAWAL_BUKA ENTER STATEMENT_METHODS
KURUNG_KURAWAL_TUTUP
STATEMENT_CLASS -> CLASS VAR KURUNG_KURAWAL_BUKA STATEMENT_METHODS ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_CLASS -> CLASS VAR KURUNG_KURAWAL_BUKA STATEMENT_METHODS
KURUNG_KURAWAL_TUTUP
STATEMENT_CLASS -> CLASS KURUNG_KURAWAL_BUKA ENTER STATEMENT_METHODS ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_CLASS -> CLASS KURUNG_KURAWAL_BUKA ENTER STATEMENT_METHODS
KURUNG_KURAWAL_TUTUP
STATEMENT_CLASS -> CLASS KURUNG_KURAWAL_BUKA STATEMENT_METHODS ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_CLASS -> CLASS KURUNG_KURAWAL_BUKA STATEMENT_METHODS
KURUNG_KURAWAL_TUTUP
STATEMENT_METHODS -> FUNC_NAME ENTER KURUNG_KURAWAL_BUKA ENTER SENTENCE ENTER
```

```
KURUNG_KURAWAL_TUTUP
STATEMENT_METHODS -> FUNC_NAME KURUNG_KURAWAL_BUKA ENTER SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_METHODS -> FUNC_NAME ENTER KURUNG_KURAWAL_BUKA SENTENCE ENTER
KURUNG_KURAWAL_TUTUP
STATEMENT_METHODS -> FUNC_NAME ENTER KURUNG_KURAWAL_BUKA ENTER SENTENCE
KURUNG_KURAWAL_TUTUP
STATEMENT_METHODS -> FUNC_NAME KURUNG_KURAWAL_BUKA SENTENCE
KURUNG_KURAWAL_TUTUP
STATEMENT_METHODS -> ENTER STATEMENT_METHODS
STATEMENT_METHODS -> STATEMENT_METHODS ENTER
STATEMENT_METHODS -> STATEMENT_METHODS ENTER STATEMENT_METHODS
STATEMENT_METHODS -> GET STATEMENT_METHODS
STATEMENT_OBJECT -> VAR DOT VAR ASSIGN VAR TITIK_KOMA
STATEMENT_OBJECT -> VAR DOT VAR ASSIGN INT TITIK_KOMA
STATEMENT_OBJECT -> VAR DOT VAR ASSIGN STRING TITIK_KOMA
STATEMENT_OBJECT -> VAR DOT STATEMENT_OBJECT
STATEMENT_OBJECT -> VAR DOT FUNC_NAME TITIK_KOMA
STATEMENT_OBJECT -> VAR DOT FUNC_NAME
STATEMENT_OBJECT -> VAR DOT VAR
STATEMENT_OBJECT -> VAR DOT VAR TITIK_KOMA
STATEMENT_OBJECT -> FUNC_NAME DOT FUNC_NAME TITIK_KOMA
STATEMENT_OBJECT -> FUNC_NAME DOT FUNC_NAME
STATEMENT_OBJECT -> VAR DOT VAR KURUNG_BUKA STATEMENT_OBJECT KURUNG_TUTUP
TITIK_KOMA
STATEMENT_STRING_KONKAT -> STRING
STATEMENT_STRING_KONKAT -> VAR
STATEMENT_STRING_KONKAT -> STRING PLUS STATEMENT_STRING_KONKAT
STATEMENT_STRING_KONKAT -> VAR PLUS STATEMENT_STRING_KONKAT
STATEMENT_IN -> VAR IN VAR
STATEMENT_IN -> VAR IN STRING
STATEMENT_IN -> INT IN STRING
STATEMENT_IN -> INT IN INT
STATEMENT_IN -> VAR IN STRING
STATEMENT_IN -> STRING IN VAR
STATEMENT_IN -> STRING IN INT
STATEMENT_IN -> INT IN VAR
DELETE_STATEMENT -> DELETE STATEMENT_OBJECT TITIK_KOMA
DELETE_STATEMENT -> DELETE STRING TITIK_KOMA
DELETE_STATEMENT -> DELETE INT TITIK_KOMA
DELETE_STATEMENT -> DELETE VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET TITIK_KOMA
DELETE_STATEMENT -> DELETE VAR TITIK_KOMA
STRING_KONKAT_SENTENCE -> VAR SUMPLUS STRING_KONKAT_SENTENCE TITIK_KOMA
STRING_KONKAT_SENTENCE -> VAR ASSIGN STRING_KONKAT_SENTENCE TITIK_KOMA
STRING_KONKAT_SENTENCE -> STRING PLUS STRING TITIK_KOMA
STRING_KONKAT_SENTENCE -> STRING PLUS STRING_KONKAT_SENTENCE
STRING_KONKAT_SENTENCE -> VAR PLUS STRING_KONKAT_SENTENCE
STRING_KONKAT_SENTENCE -> VAR
STRING_KONKAT_SENTENCE -> STRING
SWITCH_EXPR -> VAR
SWITCH_EXPR -> FUNC_NAME
CASE_EXPR -> VAR
CASE_EXPR -> INT
CASE_EXPR -> FLOAT
```

```
CASE_EXPR -> STRING
RETURN_VAR -> INT
RETURN_VAR -> STRING
RETURN_VAR -> FLOAT
RETURN_VAR -> VAR
RETURN_VAR -> BOOL
RETURN_VAR -> STATEMENT_STRING_KONKAT
RETURN_VAR -> OBJECT_SENTENCE
RETURN_VAR -> STATEMENT_ARRAY
RETURN_VAR -> ARIT_SENTENCE
RETURN_VAR -> LOGIC_SENTENCE
OBJECT_VAR -> VAR
OBJECT_VAR -> STRING
OBJECT_VAR -> INT
OBJECT_VAR -> FLOAT
OBJECT_VAR -> BOOL
ARRAY_VAR -> ARIT_VAR
ARRAY_VAR -> STRING
ARRAY_VAR -> INT
ARRAY_VAR -> BOOL
LOGIC_VAR -> BOOL
LOGIC_VAR -> NOT BOOL
LOGIC_VAR -> INT
LOGIC_VAR -> VAR
PARAMETER -> BOOL
PARAMETER -> LOGIC_SENTENCE
PARAMETER -> STRING
PARAMETER -> STATEMENT_STRING_KONKAT
PARAMETER -> ARIT_SENTENCE
PARAMETER -> VAR ASSIGN OBJECT_VAR
PARAMETER -> PARAMETER KOMA PARAMETER
PARAMETER -> ASSIGNMENT_SENTENCE
PARAMETER -> STATEMENT_INVOKE_FUNC
PARAMETER -> STATEMENT_IN
FUNC_NAME -> VAR KURUNG_BUKA KURUNG_TUTUP
FUNC_NAME -> VAR KURUNG_BUKA PARAMETER KURUNG_TUTUP
FUNC_NAME -> VAR KURUNG_BUKA FUNC_NAME KURUNG_TUTUP
FUNC_NAME -> NEW VAR KURUNG_BUKA KURUNG_TUTUP
FUNC_NAME -> NEW VAR KURUNG_BUKA PARAMETER KURUNG_TUTUP
FUNC_NAME -> NEW VAR KURUNG_BUKA FUNC_NAME KURUNG_TUTUP
LOGIC_OPERATOR -> AND
LOGIC_OPERATOR -> OR
LOGIC_OPERATOR -> LE
LOGIC_OPERATOR -> LEQ
LOGIC_OPERATOR -> NOT_EQUAL_OPERATOR
LOGIC_OPERATOR -> EQUAL_OPERATOR
LOGIC_OPERATOR -> NOT_EQUAL_TO_OPERATOR
LOGIC_OPERATOR -> EQUAL_TO_OPERATOR
LOGIC_OPERATOR -> GE
LOGIC_OPERATOR -> GEQ
LOGIC_SENTENCE -> LOGIC_VAR LOGIC_OPERATOR LOGIC_VAR
LOGIC_SENTENCE -> LOGIC_VAR LOGIC_OPERATOR LOGIC_SENTENCE
LOGIC_SENTENCE -> LOGIC_VAR
LOGIC_SENTENCE -> ARIT_SENTENCE LOGIC_OPERATOR LOGIC_VAR
```

```
LOGIC_SENTENCE -> KURUNG_BUKA ARIT_SENTENCE KURUNG_TUTUP LOGIC_OPERATOR
LOGIC_VAR
LOGIC_SENTENCE -> LOGIC_VAR EQUAL_OPERATOR NULL TITIK_KOMA
LOGIC_SENTENCE -> KURUNG_BUKA LOGIC_SENTENCE KURUNG_TUTUP
LOGIC_SENTENCE -> LOGIC_SENTENCE LOGIC_OPERATOR LOGIC_SENTENCE
LOGIC_OPERATOR -> LOGIC_VAR NOT_EQUAL_OPERATOR NULL TITIK_KOMA
CONST_VAR -> INT
CONST_VAR -> STRING
CONST_VAR -> BOOL
CONST_VAR -> CONST
CONST -> VAR
LET_VAR -> CONST_VAR
LET_VAR -> VAR
THROW_VAR -> INT
THROW_VAR -> STRING
THROW_VAR -> VAR
THROW_VAR -> FUNC_NAME
ASSIGNMENT_SENTENCE -> LET LET_VAR ASSIGN LET_VAR TITIK_KOMA
ASSIGNMENT_SENTENCE -> LET LET_VAR ASSIGN ARIT_SENTENCE TITIK_KOMA
ASSIGNMENT_SENTENCE -> LET LET_VAR ASSIGN LOGIC_SENTENCE TITIK_KOMA
ASSIGNMENT_SENTENCE -> LET LET_VAR ASSIGN NEW LET_VAR KURUNG_BUKA KURUNG_TUTUP
TITIK_KOMA
ASSIGNMENT_SENTENCE -> LET LET_VAR ASSIGN FUNC_NAME TITIK_KOMA
ASSIGNMENT_SENTENCE -> LET LET_VAR ASSIGN STATEMENT_CLASS TITIK_KOMA
ASSIGNMENT_SENTENCE -> CONST_VAR CONST_VAR ASSIGN CONST_VAR TITIK_KOMA
ASSIGNMENT_SENTENCE -> CONST_VAR CONST_VAR ASSIGN ARIT_SENTENCE TITIK_KOMA
ASSIGNMENT_SENTENCE -> CONST_VAR CONST_VAR ASSIGN LOGIC_SENTENCE TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR ASSIGN INT TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR ASSIGN BOOL TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR ASSIGN STRING TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR ASSIGN ARIT_OPERATOR TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR ASSIGN ARIT_VAR TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR ASSIGN ARIT_SENTENCE TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR ASSIGN LOGIC_SENTENCE TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR ASSIGN BOOL TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR ASSIGN STATEMENT_CLASS TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET ASSIGN
LOGIC_SENTENCE TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET ASSIGN
ARIT_SENTENCE TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET ASSIGN STRING
TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET ASSIGN INT
TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET ASSIGN FLOAT
TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET ASSIGN
STRING_KONKAT_SENTENCE TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR ASSIGN STATEMENT_ARRAY
ASSIGNMENT_SENTENCE -> LET LET_VAR ASSIGN STATEMENT_ARRAY
ASSIGNMENT_SENTENCE -> CONST CONST_VAR ASSIGN STATEMENT_ARRAY
ASSIGNMENT_SENTENCE -> CONST CONST_VAR ASSIGN STATEMENT_CLASS TITIK_KOMA
ASSIGNMENT_SENTENCE -> CONST CONST_VAR ASSIGN FUNC_NAME TITIK_KOMA
```

```
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET SUMPLUS
ARIT_SENTENCE TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET SUMPLUS INT
TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET SUMPLUS FLOAT
TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET SUMMIN
ARIT_SENTENCE TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET SUMMIN INT
TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR OPEN_BRACKET INDEX_VAR CLOSE_BRACKET SUMMIN FLOAT
TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR ASSIGN NULL
ASSIGNMENT_SENTENCE -> VAR VAR ASSIGN KURUNG_KURAWAL_BUKA OBJECT_SENTENCE
KURUNG_KURAWAL_TUTUP TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR VAR ASSIGN KURUNG_KURAWAL_BUKA ENTER
OBJECT_SENTENCE KURUNG_KURAWAL_TUTUP TITIK_KOMA
ASSIGNMENT_SENTENCE -> VAR ASSIGN KURUNG_KURAWAL_BUKA OBJECT_SENTENCE
KURUNG_KURAWAL_TUTUP TITIK_KOMA
OBJECT_SENTENCE -> OBJECT_VAR TITIK_DUA OBJECT_VAR
OBJECT_SENTENCE -> OBJECT_VAR TITIK_DUA OBJECT_VAR ENTER
OBJECT_SENTENCE -> OBJECT_SENTENCE KOMA ENTER OBJECT_SENTENCE
OBJECT_SENTENCE -> OBJECT_SENTENCE KOMA OBJECT_SENTENCE
OBJECT_SENTENCE -> OBJECT_SENTENCE KOMA ENTER OBJECT_SENTENCE KOMA
OBJECT_SENTENCE -> OBJECT_SENTENCE KOMA OBJECT_SENTENCE KOMA
INDEX_VAR -> INT
INDEX_VAR -> VAR
ARIT_VAR -> INT
ARIT_VAR -> FLOAT
ARIT_VAR -> VAR OPEN_BRACKET INT CLOSE_BRACKET
ARIT_VAR -> VAR
ARIT_VAR -> STATEMENT_OBJECT
ARIT_OPERATOR -> PLUS
ARIT_OPERATOR -> MINUS
ARIT_OPERATOR -> DIV
ARIT_OPERATOR -> MULT
ARIT_OPERATOR -> MODULO
INC_OPERATOR -> SUMPLUS
DEC_OPERATOR -> SUMMIN
INC_OPERATOR -> PLUS PLUS
DEC_OPERATOR -> MINUS MINUS
INC -> VAR INC_OPERATOR
DEC -> VAR DEC_OPERATOR
INC_VAR -> INT
DEC_VAR -> INT
INC_VAR -> VAR
DEC_VAR -> VAR
INC_DISCRETE -> INC_VAR PLUS PLUS TITIK_KOMA
DEC_DISCRETE -> DEC_VAR MINUS MINUS TITIK_KOMA
ARIT_SENTENCE -> ARIT_VAR ARIT_OPERATOR ARIT_VAR
ARIT_SENTENCE -> ARIT_VAR ARIT_OPERATOR ARIT_SENTENCE
ARIT_SENTENCE -> ARIT_VAR ARIT_OPERATOR ARIT_VAR
ARIT_SENTENCE -> ARIT_VAR ARIT_OPERATOR ARIT_SENTENCE
ARIT_SENTENCE -> KURUNG_BUKA ARIT_SENTENCE KURUNG_TUTUP
```

```
ARIT_SENTENCE -> ARIT_SENTENCE ARIT_OPERATOR ARIT_SENTENCE
ARIT_SENTENCE -> ARIT_VAR
INC_SENTENCE -> ARIT_VAR INC_OPERATOR INT TITIK_KOMA
DEC_SENTENCE -> ARIT_VAR SUMMIN INT TITIK_KOMA
INC_SENTENCE -> ARIT_VAR INC_OPERATOR
INC_SENTENCE -> ARIT_VAR INC_OPERATOR
INC -> INT INC_OPERATOR
DEC -> INT DEC_OPERATOR
SENTENCE -> STRING_KONKAT_SENTENCE
SENTENCE -> INC_DISCRETE
SENTENCE -> DELETE_STATEMENT
SENTENCE -> DEC_DISCRETE
SENTENCE -> LOGIC_SENTENCE TITIK_KOMA
SENTENCE -> ASSIGNMENT_SENTENCE
SENTENCE -> INC_SENTENCE
SENTENCE -> DEC_SENTENCE
SENTENCE -> ARIT_SENTENCE TITIK_KOMA
SENTENCE -> STATEMENT_FOR
SENTENCE -> STATEMENT_FUNC
SENTENCE -> STATEMENT_WHILE
SENTENCE -> STATEMENT_BLOCK_IF
SENTENCE -> STATEMENT_ARRAY
SENTENCE -> STATEMENT_SWITCH
SENTENCE -> STATEMENT_TRY_CATCH
SENTENCE -> STATEMENT_TRY_CATCH_FINALLY
SENTENCE -> STATEMENT_TRY_FINALLY
SENTENCE -> ENTER SENTENCE
SENTENCE -> SENTENCE ENTER SENTENCE
SENTENCE -> SENTENCE ENTER
SENTENCE -> STATEMENT_CLASS
SENTENCE -> STATEMENT_OBJECT
SENTENCE -> STATEMENT_INVOKE_FUNC
SENTENCE -> CONTINUE TITIK_KOMA
SENTENCE -> SENTENCE CONTINUE TITIK_KOMA
SENTENCE -> SENTENCE BREAK TITIK_KOMA
SENTENCE -> BREAK TITIK_KOMA
SENTENCE -> STATEMENT_RETURN
IF_SENTENCE -> SENTENCE
IF_SENTENCE -> STATEMENT_THROW
IF_SENTENCE -> CONTINUE
IF_SENTENCE -> ENTER IF_SENTENCE
IF_SENTENCE -> IF_SENTENCE ENTER IF_SENTENCE
IF_SENTENCE -> IF_SENTENCE ENTER
FUNC_SENTENCE -> SENTENCE
FUNC_SENTENCE -> STATEMENT_RETURN
FUNC_SENTENCE -> SENTENCE STATEMENT_RETURN
FOR_SENTENCE -> SENTENCE
FOR_SENTENCE -> BREAK TITIK_KOMA
FOR_SENTENCE -> SENTENCE BREAK TITIK_KOMA
FOR_SENTENCE -> CONTINUE TITIK_KOMA
FOR_SENTENCE -> SENTENCE CONTINUE TITIK_KOMA
WHILE_SENTENCE -> SENTENCE
WHILE_SENTENCE -> BREAK TITIK_KOMA
WHILE_SENTENCE -> SENTENCE BREAK TITIK_KOMA
```

```
WHILE_SENTENCE -> CONTINUE TITIK_KOMA
WHILE_SENTENCE -> SENTENCE CONTINUE TITIK_KOMA
BOOL -> TRUE
BOOL -> FALSE
NEG_INT -> MINUS INT
PLUS_INT -> PLUS INT
INT -> PLUS_INT
INT -> NEG_INT
IF -> if
ELSE -> else
WHILE -> while
DO -> do
FOR -> for
BREAK -> break
PLUS -> plus
MINUS -> minus
DIV -> div
MULT -> mult
AND -> and
OR -> or
NOT -> not
ARIT_EQUAL -> arit_equal
SUMPLUS -> sumplus
SUMMIN -> summin
SUMMULT -> summult
SUMDIV -> sumdiv
LE -> le
LEQ -> leq
GE -> ge
GEQ -> geq
EQUAL_OPERATOR -> equal_operator
NOT_EQUAL_OPERATOR -> not_equal_operator
EQUAL_TO_OPERATOR -> equal_to_operator
NOT_EQUAL_TO_OPERATOR -> not_equal_to_operator
ASSIGN -> assign
MODULO -> modulo
KOMA -> koma
TITIK_KOMA -> titik_koma
TITIK_DUA -> titik_dua
KURUNG_KURAWAL_BUKA -> kurung_kurawal_buka
KURUNG_KURAWAL_TUTUP -> kurung_kurawal_tutup
KURUNG_BUKA -> kurung_buka
KURUNG_TUTUP -> kurung_tutup
OPEN_BRACKET -> open_bracket
CLOSE_BRACKET -> close_bracket
ENTER -> enter
TRUE -> true
FALSE -> false
CONST -> const
INT -> int
CHAR -> char
LET -> let
FLOAT -> float
STRING -> string
```

```
VAR -> var
SWITCH -> switch
CASE -> case
DEFAULT -> default
TRY -> try
CATCH -> catch
RETURN -> return
FINALLY -> finally
CLASS -> class
DOT -> dot
FUNC -> func
NULL -> null
CONTINUE -> continue
THROW -> throw
NEW -> new
DELETE -> delete
GET -> get
IN -> in
```

II.III Program Parser Javascript

Program dibuat menggunakan bahasa pemrograman Python 3.

II.III.I Modul CFGtoCNF.py

Modul ini dibuat untuk merubah konfigurasi grammar CFG menjadi bentuk CNF. Module ini akan membaca *grammar* dari file CFG.txt dan akan dikonversi menjadi CNF yang berbentuk dalam struktur data *hash table* dan akan juga ditulis ke dalam bentuk CNF.txt

No.	Fungsi / Prosedur	Tujuan
1.	`readCFG`	Melakukan pembacaan file .txt yang berisi grammar dan dimasukkan ke dalam sebuah array
2.	`addRule`	Menambahkan rule yang dibaca ke variabel global yang telah didefinisikan berupa *hash table*.
3.	`CFGConverter`	Melakukan konversi grammar CFG menjadi bentuk CNF
4.	`CNFMap`	Melakukan mapping dari grammar CNF yang telah dihasilkan ke dalam bentuk dictionary agar dapat diolah dengan fungsi parser CYK yang telah dibuat
5.	`CNFWriter`	Fungsi tambahan untuk menuliskan hasil

		konversi CFG ke CNF dalam bentuk file .txt

II.III.II Modul lexer.py

Modul ini dibuat untuk membaca file yang berisi kode program JavaScript dan akan melakukan pembacaan tiap kata ataupun simbol dan diubah menjadi sebuah token.

No.	Fungsi / Prosedur	Tujuan
1.	`lex`	Mencocokkan tiap kata ataupun simbol yang terbaca dari file .js dengan aturan token yang telah dibuat dan menyimpannya ke dalam sebuah array yang berisi token
2.	`createToken`	Melakukan pembacaan terhadap file js dan mengubahnya menjadi token yang telah dibuat menggunakan fungsi `lex`

II.III.III Modul CYK.py

Modul ini digunakan untuk melakukan parsing terhadap token yang telah dibuat dan dicocokkan dengan konfigurasi CNF yang telah dihasilkan. File ini akan menghasilkan keputusan mengenai file .js yang dilakukan pengecekkan apakah sesuai dengan grammar yang telah dibuat.

No.	Fungsi / Prosedur	Tujuan
1.	`CYK`	Melakukan *parsing* terhadap token yang telah dibuat berdasarkan dictionary CNF. Dilakukan dengan algoritma CYK, dan menghasilkan keputusan apakah kode program sesuai dengan CNF atau tidak.

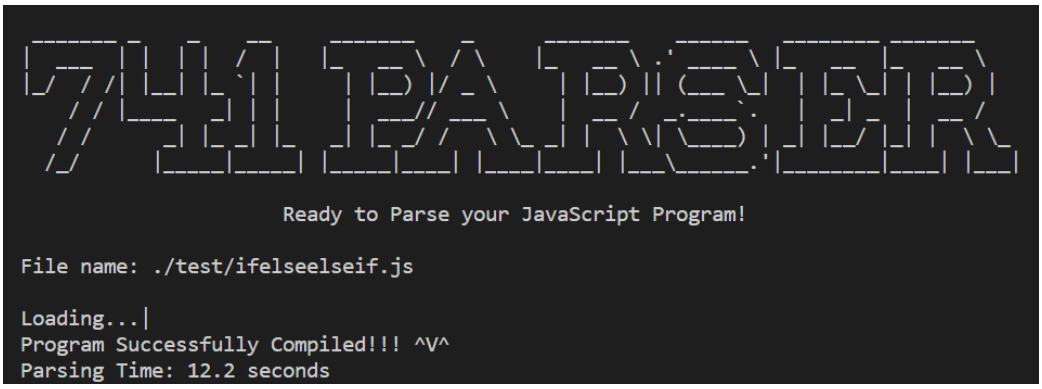
BAB III
PENGUJIAN

III.I Pengujian Conditional

Menggunakan kode Javascript,

```
let x = 0;
if(x == 0){
    console.log("x = 0");
    return 5;
}else if((x+1) == 2){
    console.log("Benar");
}else{
    console.log("Salah");
}
```

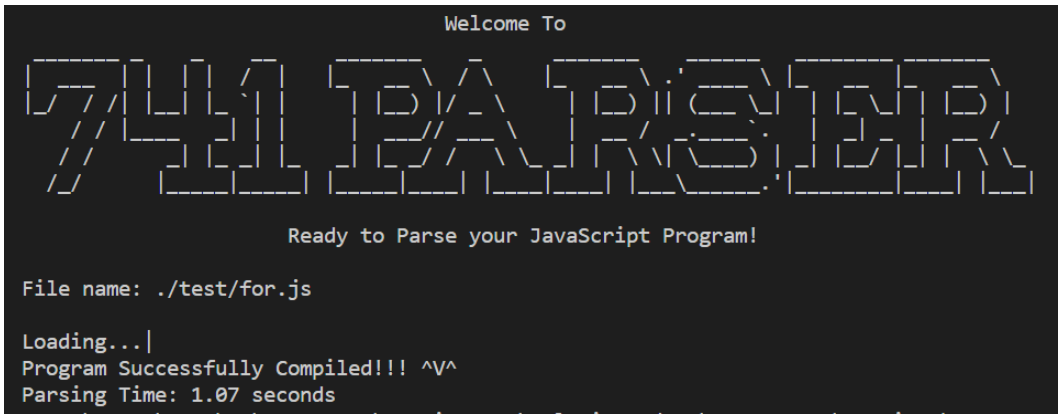
Hasilnya adalah,



III.II Pengujian Perulangan For

Menggunakan kode Javascript,

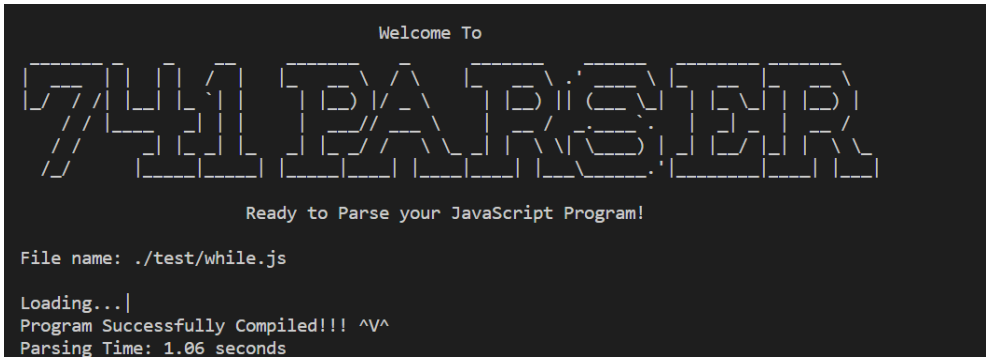
```
for(i = 0; i < 2; i++){
    console.log("A");
    break;
}
```



III.III Pengujian Perulangan While

Menggunakan kode Javascript,

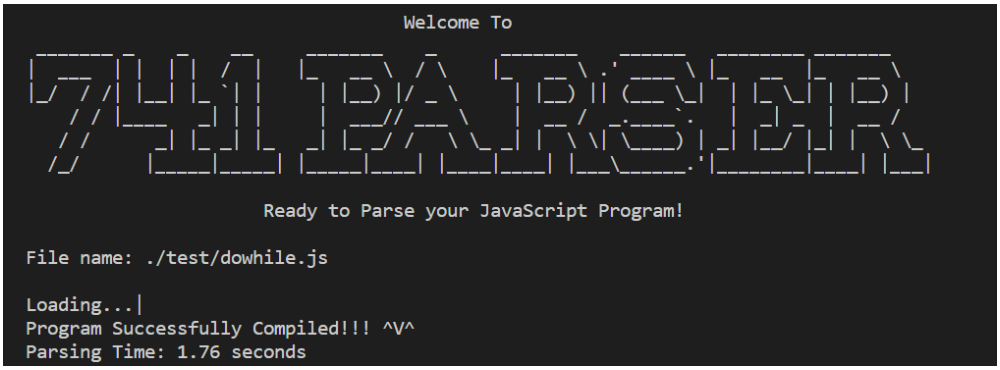
```
let x = 0;
x+=1;
while(x<10){
    this.something(x);
}
```



III.IV Pengujian Perulangan Do-While

Menggunakan kode Javascript,

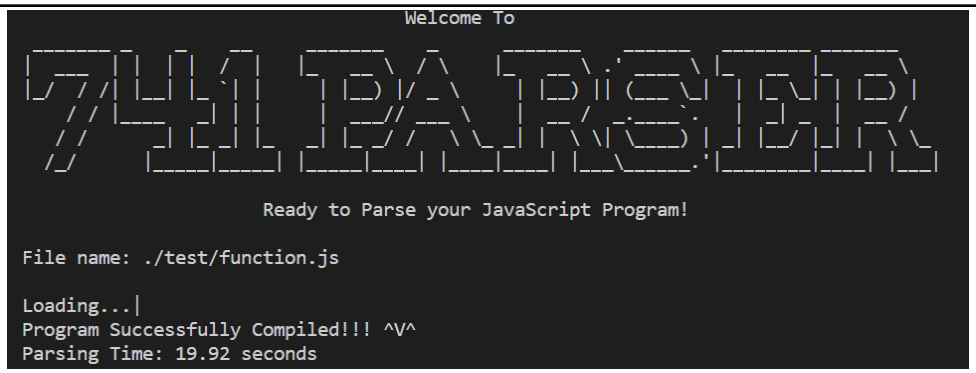
```
// do while
var i = 0;
do {
    i++;
    break;} while (i < 10);
console.log(i);
```



III.V Pengujian Function

Menggunakan kode Javascript,

```
function simpleFor(x)
{
    function fun(){
        console.log("Gak");
    }
    for (i = 0; x < 2; x++)
    {
        x = i + 2;
        console.log(i + "iterasi");
    }
    return x;
}
```

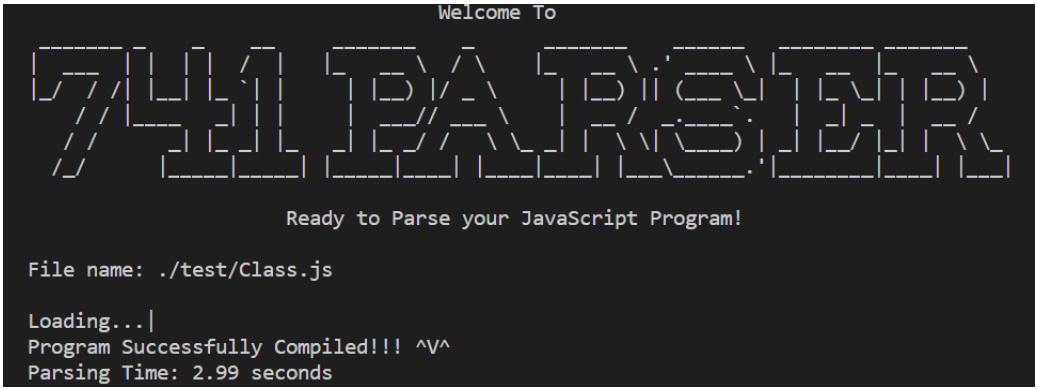


III.VI Pengujian Class

Menggunakan kode Javascript,

```
class Car {
    constructor(name, year) {
      this.name = name;
      this.year = year;
    }
```

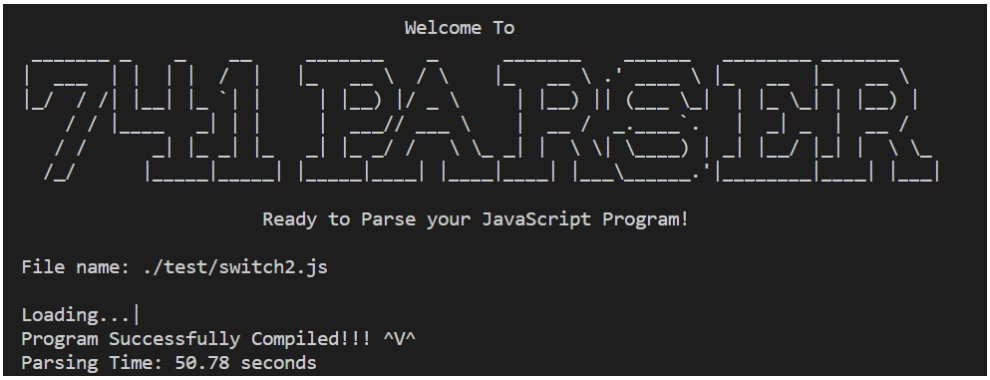
```
    age() {
        return Date;
    }
}
```



III.VII Pengujian Switch

Menggunakan kode Javascript,

```
switch (new Date()) {
    case 0:
      day = "Sunday";
      break;
    case 1:
      day = "Monday";
      break;
    case 2:
       day = "Tuesday";
      break;
    case 3:
      day = "Wednesday";
      break;
    case 4:
      day = "Thursday";
      break;
    case 5:
      day = "Friday";
      break;
    case 6:
      day = "Saturday";
  }
```



III.VIII Pengujian Array

Menggunakan kode Javascript,

```
a = [1,2,3];
a[i] = 5;
```

III.IX Pengujian Throw

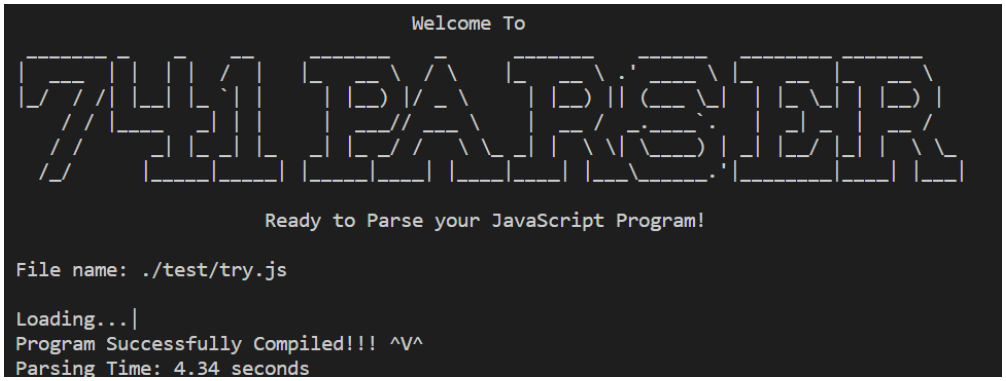
Menggunakan kode Javascript,

```
const x = 5;

if(x==5){
    throw new Error("Interupsi Woi");
}

try{
    console.log("Some");
}catch (err){
    console.log("Error Mas/Mba");
}finally{
    console.log("Lega");
}
```

III.X Pengujian Try

Menggunakan kode Javascript,

```
const word = "Hey Boy";
try{
    console.log("Some");
}catch (err){
    console.log("Error Mas/Mba");
}finally{
    console.log("Lega");
}
```



III.XI Pengujian InputAcc.js

Menggunakan kode Javascript pada spesifikasi tugas besar sebagai berikut,

```
function do_something(x) {
    // This is a sample comment
    if (x == 0) {
      return 0;
    } else if (x + 4 == 1) {
      if (true) {
```

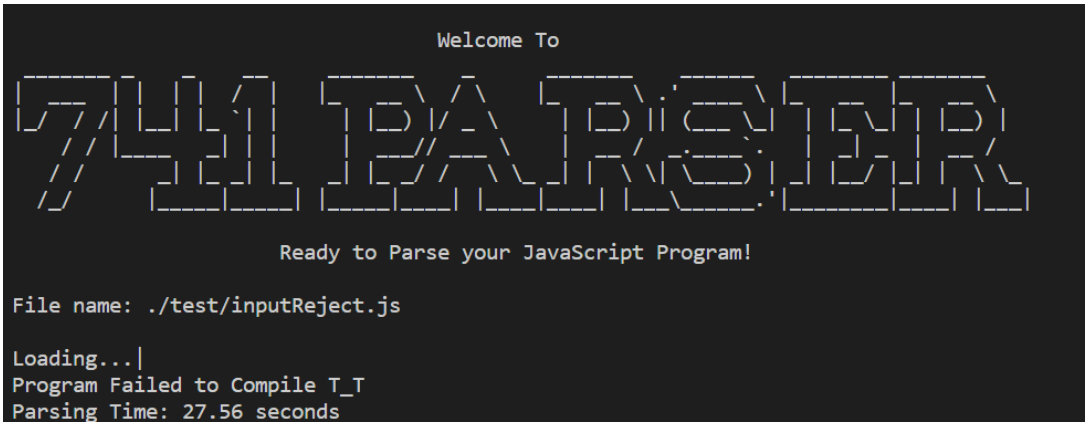
```
        return 3;
      } else {
        return 2;
      }
    } else if (x == 32) {
      return 4;
    } else {
      return "Momen";
    }
  }
```

III.XII Pengujian inputReject.js

Menggunakan kode Javascript pada spesifikasi tugas besar sebagai berikut,

```
function do_something(x) {
    // This is a sample multiline comment
    if (x == 0) {
      return 0;
    } else if x + 4 == 1 {
      if (true) {
        return 3;
      } else {
        return 2;
      }
    } else if (x == 32) {
      return 4;
    } else {
      return "Momen";
    }
  }
```

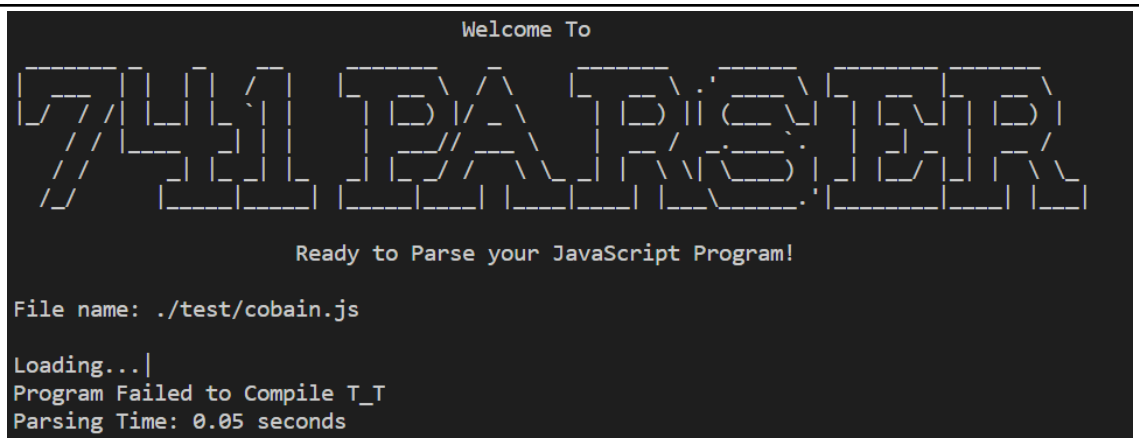
dengan kesalahan pada `x+4 ==1` karena tidak menggunakan tanda kurung, sedangkan untuk *condition* pada if, diperlukan tanda kurung. Oleh sebab itu, parser menganggap kode tidak ada dalam grammar Javascript sehingga dianggap salah.



III.XIII Pengujian Array yang Salah

Masukan kode Javascript

```
x = [[2,3];
```

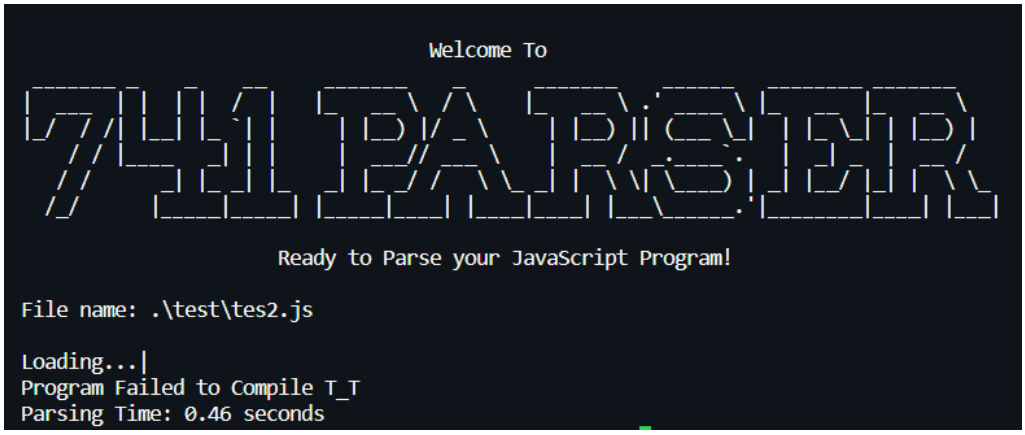


Parser menganggap kode salah karena untuk grammar array, harus diawali dan diakhiri oleh kurung siku yang saling berpasangan. Karena kurung siku pada grammar tidak lengkap, maka kode di atas dianggap tidak benar sehingga dianggap tidak ada pada grammar Javascript.

III.XIV Pengujian Block yang Tidak Sempurna

Masukan kode Javascript,

```
if(x == true || y == false){
    console.log("Sona sona");
```

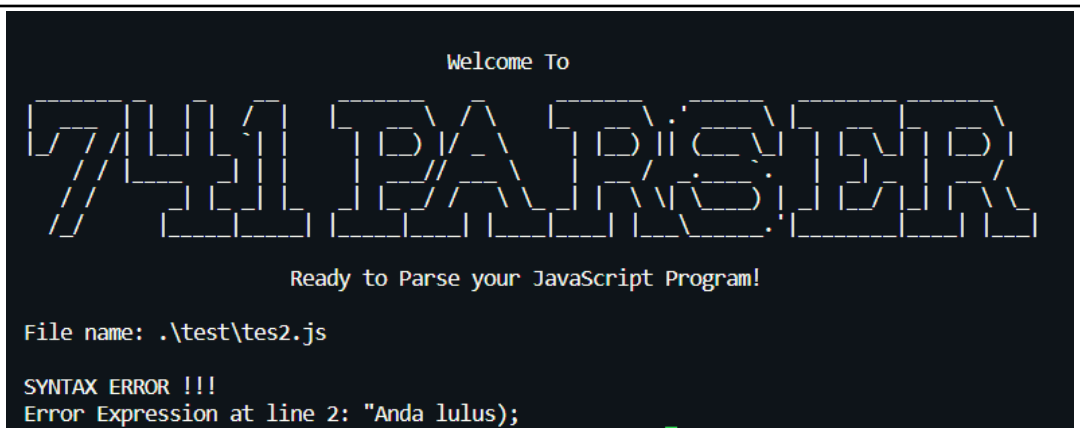


Parser menganggap program salah dikarenakan tanda kurung kurawal yang tidak lengkap. Grammar untuk if harus diakhiri dengan tanda kurung kurawal tutup. Oleh karena itu, kode di atas tidak dianggap masuk dalam grammar untuk Javascript sehingga ditolak.

III.XV Pengujian Syntax yang Tidak Sempurna

Masukan kode Javascript,

```
if(nilai_tbfo === 'A'){
    console.log("Anda lulus);
}
```



Parser menganggap program salah karena tidak ada petik pada output tidak benar. Seharusnya grammar string diawali dan diakhiri dengan tanda petik. Oleh karena itu, kode di atas tidak dianggap masuk dalam grammar untuk Javascript, sehingga ditolak.

BAB IV
SIMPULAN

IV.I Simpulan

Dari pengerjaan tugas besar ini dapat disimpulkan,

1. Parsing suatu bahasa pemrograman dapat menggunakan CFG. Terlebih, CFG digunakan untuk membuat *design* dari suatu bahasa bukan manusia
2. CYK dapat digunakan untuk menentukan apakah suatu input bahasa adalah bagian dari CFL yang dibentuk dari CFG
3. Pembentukan CNF dapat dikomputasi menggunakan algoritma perulangan yang diimplementasikan dalam bahasa Python

IV.II Saran

Saran untuk tugas besar ini adalah,

1. Membuat CFG yang lebih efektif agar waktu komputasi dapat dipersingkat. Hal ini dapat diperbaiki dengan menambah dan mendalami CFG agar dapat membuat CFG yang lebih efisien, terutama dalam mendefinisikan rekursinya
2. Membuat algoritma CYK yang lebih efektif. Pada tugas ini, kompleksitas CYK adalah $O(N^5)$

LAMPIRAN

Tautan Repository Github: https://github.com/henryanandsr/Tubes-TBFO

Pembagian Kerja Kelompok:

NIM	Nama	Tugas
13521004	Henry Anand Septian Radityo	Design FA, Grammar CFG
13521007	Matthew Mahendra	Design FA, Grammar CFG
13521024	Ahmad Nadil	Lexer, CYK, CNF