

LAPORAN TUGAS PEMROGRAMAN
IF2124 - TEORI BAHASA FORMAL DAN OTOMATA
SEMESTER I 2023/2024

Kelompok 5.K1 / budak jawir tbfo

Eduardus Alvito Kristiadi	13522004
Ibrahim Ihsan Rasyid	13522018
Rici Trisna Putra	13522026



PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG 2023

DAFTAR ISI

BAB I.....	3
BAB II.....	4
A. Context-Free Grammar.....	4
B. Pushdown Automata.....	5
C. HTML.....	6
D. GNF.....	
BAB III.....	8
A. Aplikasi PDA dalam Pengecekan Sintaks HTML.....	8
B. Desain PDA dan Hasilnya.....	8
BAB IV.....	9
A. Spesifikasi Teknis Program.....	9
B. Hasil Pengujian Program.....	9
BAB V.....	10
A. Kesimpulan.....	10
B. Saran.....	10
C. Refleksi.....	10
DAFTAR PUSTAKA.....	11
LAMPIRAN.....	12

BAB I

DESKRIPSI MASALAH

HTML (Hypertext Markup Language) adalah bahasa markup yang digunakan untuk membuat struktur dan tampilan konten web. HTML adalah salah satu bahasa utama yang digunakan dalam pengembangan web dan digunakan untuk menggambarkan bagaimana elemen-elemen konten, seperti teks, gambar, tautan, dan media, akan ditampilkan di browser web. Setiap dokumen HTML dimulai dengan elemen `<html>`, lalu diikuti dengan `<head>` (untuk metadata dan tautan ke file eksternal) dan `<body>` (untuk konten yang akan ditampilkan)

HTML menggunakan elemen-elemen (*tags*) untuk mengelompokkan dan mengatur konten. Contohnya, `<p>` digunakan untuk paragraf teks, `<h1>` hingga `<h6>` digunakan untuk judul, `<a>` untuk tautan, `` untuk gambar, dan sebagainya. Elemen HTML sering memiliki atribut yang memberikan informasi tambahan tentang elemen tersebut. Contohnya adalah atribut `src` untuk gambar, `href` untuk tautan, dan `class` untuk memberikan elemen kelas CSS.

Sama seperti bahasa pada umumnya, HTML juga memiliki sintaks tersendiri dalam penulisannya yang dapat menimbulkan error jika tidak dipenuhi. Meskipun web browser modern seperti Chrome dan Firefox cenderung tidak menghiraukan error pada HTML, memastikan bahwa HTML benar dan terbentuk dengan baik masih penting untuk beberapa alasan seperti *Search Engine Optimization (SEO)*, aksesibilitas, *maintenance* yang lebih baik, kecepatan render, dan profesionalisme.

Dibutuhkan sebuah program pendeteksi *error* untuk HTML. Oleh sebab itu, akan diimplementasikan sebuah program yang dapat memeriksa kebenaran HTML dari segi nama *tag* yang digunakan serta *attribute* yang dimilikinya. Pada tugas pemrograman ini, digunakan konsep Pushdown Automata (PDA) dalam mencapai hal tersebut yang diimplementasikan dalam bahasa Python.

Mengingat banyaknya jenis *tags* beserta *attributes* yang tersedia pada HTML, *scope* tugas pemrograman akan dibatasi. Batasan-batasan dari *tags* dan *attributes* yang diperiksa dapat dilihat pada bab selanjutnya.

BAB II

LANDASAN TEORI

A. Context-Free Grammar

Tata Bahasa Bebas Konteks (CFG) adalah sekumpulan berhingga variabel (non terminal) dalam aturan produksi, masing-masing merepresentasikan bahasa secara rekursif atau dalam bentuk lain. Terdapat 4 tupel (benda) yang membentuk CFG yaitu [2] $G = \{VN, VT, P, S\}$ dimana VN himpunan variabel atau non terminal, VT himpunan simbol terminal, P adalah kumpulan berhingga aturan produksi (bagaimana untai-untai dalam masing-masing kelas dibangun) dan bentuk umum aturan produksinya $\alpha \rightarrow \beta$, $\alpha \in VN$, $\beta \in (VN \cup VT)^*$, dan S adalah variabel special yang disebut simbol permulaan ($S \in VN$). α berupa satu variabel, tanda “ \rightarrow ” artinya penurunan, dan β artinya untai atau string. Simbol-simbol yang digunakan antara lain[1]: (1). Karakter khusus (a, b, c,...,z, 0, 1, 2...,9 menyatakan terminal), (2). karakter A,B,C,...,Z menyatakan non terminal dan S simbol awal(mula), dan (3). String yang terdiri dari simbol-simbol dalam terminal.

Pada dasarnya, setiap aturan produksi terdiri dari simbol non-terminal (variabel) dan deretan simbol terminal (simbol akhir). CFG digunakan secara luas dalam pengembangan kompilator, analisis sintaksis, dan pemrosesan bahasa alami. Sebuah CFG terdiri dari empat komponen utama:

- Simbol Terminal (T): Simbol-simbol dasar yang merupakan bagian dari bahasa yang dihasilkan. Ini adalah simbol-simbol yang tidak dapat diperluas lebih lanjut.
- Simbol Non-Terminal (N): Variabel-variabel yang dapat diperluas oleh aturan produksi untuk membentuk rangkaian simbol terminal. Simbol non-terminal dapat dianggap sebagai placeholder untuk satu atau lebih simbol terminal.
- Aturan Produksi (P): Aturan-aturan yang mendefinisikan bagaimana simbol-simbol non-terminal dapat diperluas menjadi rangkaian simbol terminal dan/atau simbol non-terminal lainnya.
- Simbol Awal (S): Simbol non-terminal yang merupakan titik awal dari pembentukan suatu kalimat.

B. Pushdown Automata

Pushdown Automaton (PDA) adalah model automata yang secara formal menggambarkan komputasi dengan menggunakan tumpukan (stack) sebagai memori tambahan. PDA adalah perluasan dari finite automaton dengan tambahan tumpukan, yang memberikan kemampuan untuk mengenali bahasa-bahasa konteks bebas (context-free languages). PDA terdiri dari beberapa komponen, termasuk keadaan (states), simbol input, simbol tumpukan, fungsi transisi, dan simbol akhir. Berikut adalah komponen-komponen utama dari PDA:

1. States (Keadaan): Kumpulan keadaan yang mungkin, termasuk keadaan awal dan keadaan akhir.
2. Input Symbols (Simbol Input): Kumpulan simbol yang dapat dibaca oleh PDA dari input.
3. Stack Symbols (Simbol Tumpukan): Kumpulan simbol yang dapat ditempatkan atau dihapus dari tumpukan.
4. Transition Function (Fungsi Transisi): Menunjukkan cara PDA berpindah dari satu keadaan ke keadaan lainnya berdasarkan simbol input yang dibaca dan simbol tumpukan saat ini.
5. Start State (Keadaan Awal): Keadaan awal dari PDA.
6. Accept States (Keadaan Penerimaan): Keadaan-keadaan di mana PDA mengakui input dan berhenti.
7. Stack Bottom Symbol (Simbol Paling Bawah Tumpukan): Simbol khusus yang menunjukkan bagian bawah dari tumpukan.

Sebuah Pushdown Automaton (PDA) dapat didefinisikan sebagai berikut:

- Q adalah kumpulan states.
- Σ adalah kumpulan simbol input.
- Γ adalah kumpulan simbol stack (yang dapat dipush dan dipop dari stack).
- q_0 adalah keadaan awal.
- Z adalah simbol stack awal (yang pada awalnya ada di dalam stack).
- F adalah kumpulan final state.

- δ adalah fungsi transisi yang memetakan $Q \times \{\Sigma \cup \epsilon\} \times \Gamma$ ke dalam $Q \times \Gamma^*$. Dalam suatu keadaan tertentu, PDA akan membaca simbol input dan simbol stack (pada bagian atas tumpukan) dan beralih ke keadaan baru serta mengubah simbol stack.

C. HTML

HTML, atau Hypertext Markup Language, merupakan bahasa markah standar yang digunakan untuk membuat dan menyusun konten pada halaman web. HTML bersifat markup language, yang berarti ia menggunakan tag atau penanda untuk menentukan bagian-bagian tertentu dari halaman web. Hal ini memungkinkan browser untuk menginterpretasikan dan menampilkan konten secara sesuai.

Berikut adalah beberapa konsep dasar dan elemen penting terkait HTML:

1. Struktur Dasar HTML:

- Setiap halaman HTML dimulai dengan elemen `<html>` dan diakhiri dengan `</html>`.
- Bagian utama dari halaman web ditempatkan di antara elemen `<body>` dan `</body>`.

2. Tag HTML:

- Tag HTML adalah elemen-elemen dasar dalam HTML yang digunakan untuk merinci struktur dokumen.
- Contoh tag: `<p>` untuk paragraf, `<h1>` hingga `<h6>` untuk judul dengan tingkat kepentingan yang berbeda, `<a>` untuk hyperlink, `` untuk gambar, dll.

3. Atribut:

- Tag HTML dapat memiliki atribut yang memberikan informasi tambahan tentang elemen tersebut.
- Contoh: `Link ke Example` di mana href adalah atribut yang menentukan URL tujuan hyperlink.

4. Versi HTML:

- HTML telah berkembang dari versi ke versi. Versi terakhir saat ini adalah HTML5, yang diperkenalkan pada tahun 2014.

- b. HTML5 menyertakan banyak perubahan dan tambahan baru, termasuk elemen-elemen baru, dukungan multimedia, dan peningkatan kemampuan untuk pengembangan web responsif.

5. Element Konten:

- a. Berbagai tag HTML digunakan untuk menyusun konten pada halaman web, seperti teks, gambar, tabel, formulir, video, audio, dan lainnya.
- b. Tag seperti `<div>` digunakan untuk mengelompokkan elemen-elemen secara bersamaan.

6. Web Browser:

- a. Halaman HTML dapat diakses dan ditampilkan oleh web browser seperti Google Chrome, Mozilla Firefox, dan lainnya.
- b. Browser membaca dan menafsirkan HTML untuk menampilkan konten yang dapat diakses oleh pengguna.

7. Pembaruan dan Spesifikasi:

- a. World Wide Web Consortium (W3C) bertanggung jawab atas pengembangan dan pembaruan spesifikasi HTML.
- b. HTML diusahakan agar tetap menjadi standar terbuka dan dapat diakses oleh berbagai platform dan perangkat.

HTML merupakan dasar dari sebagian besar situs web di internet. Sementara HTML menyusun struktur dan konten dasar, CSS (Cascading Style Sheets) dan JavaScript sering digunakan bersama untuk menyempurnakan presentasi visual dan menambahkan interaktivitas pada halaman web.

D. Greibach Normal Form (GNF)

GNF (Greibach Normal Form) adalah bentuk normal Greibach untuk Context Free Grammar (CFG). Sebuah CFG dikatakan berada dalam GNF jika semua aturan produksi memenuhi salah satu kondisi berikut:

- Simbol awal menghasilkan ϵ . Contohnya, $S \rightarrow \epsilon$.
- Simbol non-terminal menghasilkan terminal. Contohnya, $A \rightarrow a$.
- Simbol non-terminal menghasilkan terminal yang diikuti oleh sejumlah non-terminal. Contohnya, $S \rightarrow aASB$.

Terdapat beberapa langkah untuk mengonversi CFG menjadi GNF:

- Langkah 1: Konversi tata bahasa menjadi CNF.

Jika tata bahasa yang diberikan tidak dalam CNF, konversi menjadi CNF. Anda dapat merujuk pada topik berikut untuk mengonversi CFG menjadi CNF (Chomsky Normal Form)

- Langkah 2: Jika tata bahasa mengandung rekursi kiri, hapuslah.

Jika tata bahasa bebas konteks mengandung rekursi kiri, hapuslah. Anda dapat merujuk pada topik berikut untuk menghilangkan rekursi kiri: Rekursi Kiri

- Langkah 3: Dalam tata bahasa, konversi aturan produksi yang diberikan menjadi bentuk GNF.

Jika aturan produksi apa pun dalam tata bahasa tidak dalam bentuk GNF, lakukan konversi.

Dengan melakukan langkah-langkah tersebut, dapat dilakukan konversi CFG ke dalam bentuk GNF.

BAB III

ANALISIS PEMECAHAN MASALAH

A. Aplikasi PDA dalam Pengecekan Sintaks HTML

Pada pengerjaan tugas besar ini, digunakan PDA untuk melakukan pengecekan sintaks HTML. PDA dipilih karena memiliki kemampuan untuk mengenali bahasa yang dihasilkan oleh tata bahasa tak terbatas (unrestricted grammar), yang mencakup bahasa HTML. HTML adalah contoh tata bahasa tak terbatas, dan PDA dapat digunakan untuk memodelkan struktur hierarki dan susunan tanda-tanda (tag) dalam HTML.

Pada PDA, stack digunakan sebagai struktur data utama untuk melacak tumpukan elemen atau tag HTML yang dibuka dan belum ditutup. Setiap kali membaca token HTML baru, PDA memutuskan apakah token tersebut sesuai dengan aturan penulisan HTML dan memperbarui stack sesuai.

Selain itu, PDA melakukan transisi antar keadaan (state) berdasarkan token yang sedang dibaca dan simbol pada bagian atas stack. Ini mencerminkan bagaimana PDA "mengerti" konteks saat ini dan menyesuaikan perilakunya sesuai dengan token dan tumpukan saat ini.

Setelah membaca seluruh input (token HTML), PDA memeriksa apakah dalam keadaan akhir (final state) dan tumpukan kosong. Ini menunjukkan bahwa input HTML telah diuraikan sesuai dengan sintaks yang benar.

FileHandler digunakan untuk membaca file yang berisi definisi PDA dan produksi-produksinya. Ini termasuk informasi tentang simbol-simbol, keadaan, tumpukan awal, dan aturan produksi yang digunakan oleh PDA.

FileHandler juga bertanggung jawab untuk memarsing konten file ke dalam struktur data yang dapat digunakan oleh PDA. Informasi ini kemudian digunakan oleh PDA dalam proses komputasinya.

Dalam pengecekan sintaks HTML, juga digunakan tokenizer dengan cara PDA membaca token HTML satu per satu dan memutuskan transisi berdasarkan token tersebut. Setiap transisi memperbarui keadaan dan stack sesuai dengan aturan-aturan yang didefinisikan dalam automata file. PDA juga memberikan output yang memungkinkan untuk melacak langkah-langkah transisi dan status stack. Ini dapat digunakan untuk debugging dan analisis lebih lanjut saat terjadi kesalahan atau untuk memahami bagaimana PDA membaca dan memproses input HTML.

B. Desain PDA dan Hasilnya

BAB IV

IMPLEMENTASI DAN UJI COBA

A. Spesifikasi Teknis Program

Dalam pembuatan tugas besar ini, program utama menggunakan bahasa Python. Secara keseluruhan, isi program ini meliputi parser, tokenizer, CFG, GNF

1. file_processing.py

```
2 import sys
3 from vocabulary import token_exp
4 import linecache
5
6 Codeium: Refactor | Explain | Generate Docstring
7 def lexer(filename, teks, token_exp):
8     pos = 0 # posisi karakter pada seluruh potongan teks (absolut)
9     tokens = []
10    while pos < len(teks):
11        match = None
12        for t in token_exp:
13            pattern, tag = t
14            regex = re.compile(pattern)
15            match = regex.match(teks, pos)
16            if match:
17                # !!!! DEBUGGING TEST
18                # print(match)
19                # print(pattern, tag)
20                if tag:
21                    token = tag
22                    tokens.append(token)
23                break
24        if not match:
25            temp = pos
26            while (teks[temp-1] != '\n'):
27                temp -= 1
28            line = teks[0:temp].count('\n') + 1
29            linetext = linecache.getline(filename, line)
30            print("Error in line " + str(line))
31            print(linetext)
32            sys.exit(1)
33        else:
34            pos = match.end(0)
35    return tokens
36
37 Codeium: Refactor | Explain | Generate Docstring
38 def create_token(sentence):
39     file = open(sentence)
40     char = file.read()
41     file.close()
42
43     tokens = lexer(sentence, char, token_exp)
44     tokenArray = []
45     for token in tokens:
46         tokenArray.append(token)
47     # print("TOKEN : " + " ".join(tokenArray))
48     return tokenArray
```

Pada program file_processing.py, terdapat fungsi lexer yang mengandung parameter filename, teks, dan token. Filename berisi nama berkas yang sedang diproses, teks berisi isi teks dari file tersebut, serta token_exp berisi regular expression yang mendefinisikan token. Fungsi tersebut bertugas untuk mencocokkan pola pada regex.

Apabila terdapat kecocokan, tag(token) akan masuk ke list tokens. Jika tidak, maka akan print error message. Terdapat pula fungsi create_token yang berfungsi open file dan read file.

2. FileHandler.py

```
3 class FileHandler:
4
5     Codeium: Refactor | Explain | Generate Docstring
6     def __init__(self):
7         pass
8
9     Codeium: Refactor | Explain | Generate Docstring
10    def readFile(self, filePath):
11        lines = []
12        if os.path.isfile(filePath):
13            try:
14                with open(filePath) as file:
15                    lines = [line.rstrip() for line in file]
16            except IOError as e:
17                print("File could not be opened.")
18                exit(0)
19        else:
20            print('{} :File was not found in the specified path.'.format(filePath))
21            exit(0)
22        return lines
23
24    Codeium: Refactor | Explain
25    def parseFile(self, lines):
26        '''
27        Line 1: Total States
28        Line 2: Input Word Symbols
29        Line 3: Stack Symbols
30        Line 4: Initial State Symbol
31        Line 5: Initial Stack Symbol
32        Line 6: List of Final States
33        Line 7 and onwards: Productions in form of
34        (Current State, Current Input Symbol, Current Top of Stack, Next State, Push/Pop Operation Symbol)
35        '''
36        states = lines[0].rstrip().split()
37        input_symbols = lines[1].rstrip().split()
38        stack_symbols = lines[2].rstrip().split()
39        initial_state = lines[3].rstrip().split()[0]
40        initial_stack = lines[4][0]
41        final_states = lines[5].rstrip().split()
42        productions = lines[6:]
43        for i in range(len(productions)):
44            productionku = productions[i].rstrip().split()
45            productions[i] = [productionku[0], productionku[1], productionku[2], productionku[3], [productionku[4:]]]
46        print("List index hapus")
47        listindexhapus = []
48        for i in range(0, len(productions)):
49            production = productions[i]
50            for j in range(i+1, len(productions)):
51                production2 = productions[j]
52                if(production[0] == production2[0] and production[1] == production2[1] and production[2] == production2[2] and production[3] not in production2[4]):
```

Pada program FileHandler.py, terdapat dua fungsi, yaitu Read File dan Parse File. Pada Read File, file akan dibaca dan dimasukkan ke dalam list 'lines'. Sedangkan pada fungsi parseFile, fungsi ini menghasilkan dictionary parsedLines yang berisi informasi tentang PDA, seperti states, input_symbols, stack_symbols, initial_state, initial_stack, final_states, dan productions.

3. main_PDA.txt

```
1 Q Q2 Q3
2 HTML HTMLCLOSE HEAD HEADCLOSE BODY BODYCLOSE TITLE TITLECLOSE SCRIPT SCRIPTCLOSE H1 H1CLOSE H2 H2CLOSE H3 H3CLOSE H4 H4CLOSE H5 H5CLOSE H6 H6CLOSE P PCLOSE EM ENCLOSE B BCLOSE ABBR ABBRCLOSE
3 STRONG STRONGCLOSE SMALL SMALLCLOSE DIV DIVCLOSE A ACLOSE BUTTON BUTTONCLOSE FORM FORMCLOSE TABLE TABLECLOSE TR TRCLOSE TD TDCLOSE TH THCLOSE TEXT LINK BR HR IMG INPUT < > ID STYLE CLASS REL
4 HREF SRC ALT TYPEBUTTON ACTION METHOD COMMENT
5
6 S HTMLVAR KSKA MAIN HTMLCLOSETAG GLOBALATTR HTMLCLOSETAG COMMENT CHTS HTMLTAG
7
8 Q
9 Z
10 Q3
11 Q e Z Q2 S Z
12
13 Q2 < S Q2 HTMLVAR GLOBALATTR KSKA MAIN HTMLCLOSETAG
14 Q2 < S Q2 HTMLVAR KSKA HTMLCLOSETAG
15 Q2 < HTMLTAG Q2 HTMLVAR KSKA
16 Q2 < HTMLCLOSETAG Q2 HTMLCLOSESEVAR KSKA
17 Q2 HTML HTMLVAR Q2 e
18 Q2 > KSKA Q2 e
19 Q2 HTMLCLOSE HTMLCLOSESEVAR Q2 e
20 Q2 e Z Q2 J
```

File tersebut merupakan file PDA yang akan dibaca oleh program. File tersebut mengandung elemen-elemen dari HTML yang membentuk PDA.

B. Hasil Pengujian Program

```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 3\TBFO Automata - P Judi\tubes\TBTFBO> py main.py
TOKEN : HTML -> HTMLCLOSE
Enter the automata file path: pda_rici_2.txt
Reading Automata File
Loading Details from Automata File:
List index hapus
[[['HTMLVAR', 'GLOBALATTR', 'KSKA', 'MAIN', 'HTMLCLOSETAG']]
[['HTMLVAR', 'KSKA', 'HTMLCLOSETAG']]
[3]
[['Q', 'e', 'Z', 'Q2', [['S', 'Z']]], ['Q2', 'c', 'S', 'Q2', [['HTMLVAR', 'GLOBALATTR', 'KSKA', 'MAIN', 'HTMLCLOSETAG'], ['HTMLVAR', 'KSKA', 'HTMLCLOSETAG']]], ['Q2', 'c', 'S', 'Q2', [['HTMLVAR', 'KSKA', 'HTMLCLOSETAG']]], ['Q2', 'c', 'HTMLTAG', 'Q2', [['HTMLVAR', 'KSKA']]], ['Q2', 'c', 'HTMLCLOSETAG', 'Q2', [['HTMLCLOSEVAR', 'KSKA']]], ['Q2', 'HTML', 'HTMLVAR', 'Q2', [['e']]], ['Q2', '>', 'KSKA', 'Q2', [['e']]], ['Q2', 'HTMLCLOSE', 'HTMLCLOSEVAR', 'Q2', [['e']]], ['Q2', 'e', 'Z', 'Q3', [['e']]]]
States: ['Q', 'Q2', 'Q3']
Input Symbols: ['HTML', 'HTMLCLOSE', 'HEAD', 'HEADCLOSE', 'BODY', 'BODYCLOSE', 'TITLE', 'TITLECLOSE', 'SCRIPT', 'SCRIPTCLOSE', 'H1', 'H1CLOSE', 'H2', 'H2CLOSE', 'H3', 'H3CLOSE', 'H4', 'H4CLOSE', 'H5', 'H5CLOSE', 'H6', 'H6CLOSE', 'P', 'PCLOSE', 'EM', 'EMCLOSE', 'B', 'BCLOSE', 'ABBR', 'ABBRCLOSE', 'STRONG', 'STRONGCLOSE', 'SMALL', 'SMALLCLOSE', 'DIV', 'DIVCLOSE', 'A', 'ACLOSE', 'BUTTON', 'BUTTONCLOSE', 'FORM', 'FORMCLOSE', 'TABLE', 'TACLOSE', 'TR', 'TRCLOSE', 'TD', 'TDCLOSE', 'TH', 'THCLOSE', 'TEXT', 'LINK', 'BR', 'HR', 'IMG', 'INPUT', '<', '>', 'ID', 'STYLE', 'CLASS', 'REL', 'HREF', 'SRC', 'ALT', 'TYPEBUTTON', 'TYPEINPUT', 'ACTION', 'METHOD', 'CO', 'MENEM']
Stack Symbols: ['S', 'HTMLVAR', 'KSKA', 'MAIN', 'HTMLCLOSETAG', 'GLOBALATTR', 'HTMLCLOSETAG', 'COMMENT', 'CHTS', 'HTMLTAG']
Initial State: Q
Initial Stack Symbol: Z
Final States: ['Q3']
Productions List:
[['Q', 'e', 'Z', 'Q2', [['S', 'Z']]]
[['Q2', 'c', 'S', 'Q2', [['HTMLVAR', 'GLOBALATTR', 'KSKA', 'MAIN', 'HTMLCLOSETAG'], ['HTMLVAR', 'KSKA', 'HTMLCLOSETAG']]]
[['Q2', 'c', 'HTMLTAG', 'Q2', [['HTMLVAR', 'KSKA']]]
[['Q2', 'c', 'HTMLCLOSETAG', 'Q2', [['HTMLCLOSEVAR', 'KSKA']]]
[['Q2', 'HTML', 'HTMLVAR', 'Q2', [['e']]]
[['Q2', '>', 'KSKA', 'Q2', [['e']]]
[['Q2', 'HTMLCLOSE', 'HTMLCLOSEVAR', 'Q2', [['e']]]
[['Q2', 'e', 'Z', 'Q3', [['e']]]]
Automata File Successfully Read
Enter test file path (or end): test_b.txt
TOKEN : HTML -> HTMLCLOSE
Reading string from file...
String successfully read
['HTML', '>', 'HTMLCLOSE']
Computing the Transition Table:
State Input Stack Move
Q - Z (Z, ['Z'])
Q e [['S', 'Z']] (Q2, ['Z'])
Q2 HTML ['S', 'Z'] (Q2, [['S', 'Z']])
Q2 > ['S', 'Z'] (Q2, [['S', 'Z']])
Q2 HTMLCLOSE ['S', 'Z'] (Q2, [['S', 'Z']])
Q2 e ['S', 'Z'] (Q2, [['S', 'Z']])
String rejected by PDA.
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 3\TBFO Automata - P Judi\tubes\TBTFBO>
```

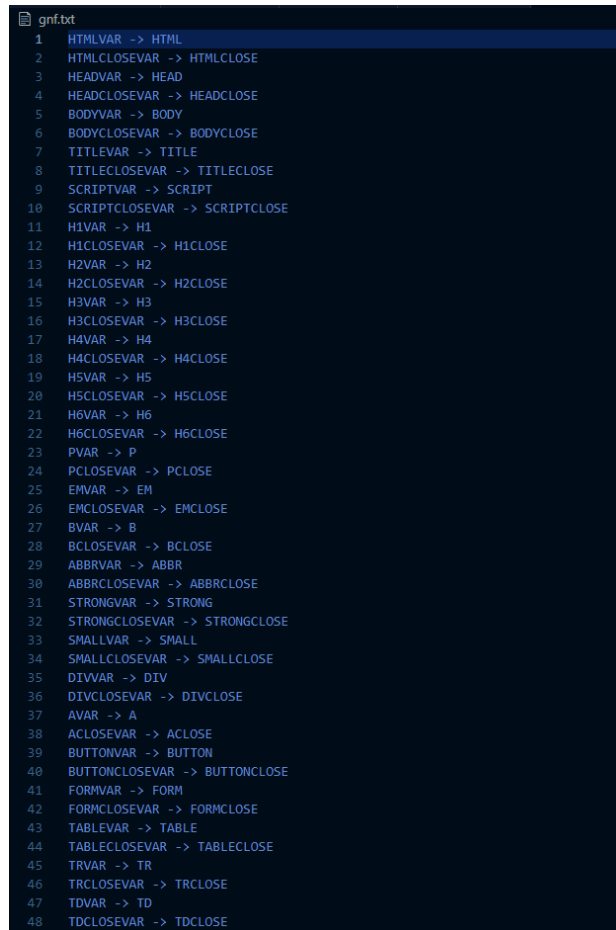
Program telah diuji untuk membaca PDA dari file pda_rici_2.txt. Tercatat states, input symbols, stack symbols, initial state, initial state symbol, final states, dan production lists dari pembacaan PDA.

C. Cfg.txt

```
cfg.txt
1 S -> HTMLTAG MAIN HTMLCLOSETAG
2 S -> CHTS HTMLTAG MAIN HTMLCLOSETAG
3 S -> HTMLTAG MAIN HTMLCLOSETAG CHTS
4 S -> CHTS HTMLTAG MAIN HTMLCLOSETAG CHTS
5 CHTS -> COMMENT CHTS
6 CHTS -> COMMENT
7 MAIN -> HEADELEMENT BODYELEMENT
8 MAIN -> CHTS HEADELEMENT BODYELEMENT
9 MAIN -> HEADELEMENT CHTS BODYELEMENT
10 MAIN -> HEADELEMENT BODYELEMENT CHTS
11 MAIN -> CHTS HEADELEMENT CHTS BODYELEMENT
12 MAIN -> HEADELEMENT CHTS BODYELEMENT CHTS
13 MAIN -> CHTS HEADELEMENT CHTS BODYELEMENT CHTS
14 HEADELEMENT -> HEADTAG ELEMENTHEAD HEADCLOSETAG
15 HEADELEMENT -> HEADTAG HEADCLOSETAG
16 BODYELEMENT -> BODYTAG ELEMENTBODY BODYCLOSETAG
17 BODYELEMENT -> BODYTAG BODYCLOSETAG
18 GLOBALATTR -> GLOBALATTR GLOBALATTR
19 GLOBALATTR -> ID
20 GLOBALATTR -> CLASS
21 GLOBALATTR -> STYLE
22 LINKATTR -> REL
23 LINKATTR -> REL LINKATTRPTIONAL
24 LINKATTR -> LINKATTRPTIONAL REL LINKATTRPTIONAL
25 LINKATTR -> LINKATTRPTIONAL REL
26 LINKATTRPTIONAL -> LINKATTRPTIONAL LINKATTRPTIONAL
27 LINKATTRPTIONAL -> HREF
28 LINKATTRPTIONAL -> ID
29 LINKATTRPTIONAL -> STYLE
30 LINKATTRPTIONAL -> CLASS
31 SCRIPTATTR -> SCRIPTATTR SCRIPTATTR
32 SCRIPTATTR -> SRC
33 SCRIPTATTR -> ID
34 SCRIPTATTR -> STYLE
35 SCRIPTATTR -> CLASS
36 AATTR -> AATTR AATTR
37 AATTR -> HREF
38 AATTR -> ID
39 AATTR -> STYLE
40 AATTR -> CLASS
41 INGATTR -> SRC
42 INGATTR -> SRC INGATTRPTIONAL
43 INGATTR -> INGATTRPTIONAL SRC INGATTRPTIONAL
44 INGATTR -> INGATTRPTIONAL SRC
45 INGATTRPTIONAL -> INGATTRPTIONAL INGATTRPTIONAL
46 INGATTRPTIONAL -> ALT
47 INGATTRPTIONAL -> ID
48 INGATTRPTIONAL -> STYLE
```

Pada pembuatan tugas besar ini, juga telah dibuat file cfg.txt untuk mempermudah pembuatan dan pembacaan PDA, di mana cfg ini digunakan untuk menghasilkan gnf.

D. Gnf.txt



```
gnf.txt
1  HTMLVAR -> HTML
2  HTMLCLOSEVAR -> HTMLCLOSE
3  HEADVAR -> HEAD
4  HEADCLOSEVAR -> HEADCLOSE
5  BODYVAR -> BODY
6  BODYCLOSEVAR -> BODYCLOSE
7  TITLEVAR -> TITLE
8  TITLECLOSEVAR -> TITLECLOSE
9  SCRIPTVAR -> SCRIPT
10 SCRIPTCLOSEVAR -> SCRIPTCLOSE
11 H1VAR -> H1
12 H1CLOSEVAR -> H1CLOSE
13 H2VAR -> H2
14 H2CLOSEVAR -> H2CLOSE
15 H3VAR -> H3
16 H3CLOSEVAR -> H3CLOSE
17 H4VAR -> H4
18 H4CLOSEVAR -> H4CLOSE
19 H5VAR -> H5
20 H5CLOSEVAR -> H5CLOSE
21 H6VAR -> H6
22 H6CLOSEVAR -> H6CLOSE
23 PVAR -> P
24 PCLOSEVAR -> PCLOSE
25 EMVAR -> EM
26 EMCLOSEVAR -> EMCLOSE
27 BVAR -> B
28 BCLOSEVAR -> BCLOSE
29 ABBRVAR -> ABBR
30 ABBRCLOSEVAR -> ABBRCLOSE
31 STRONGVAR -> STRONG
32 STRONGCLOSEVAR -> STRONGCLOSE
33 SMALLVAR -> SMALL
34 SMALLCLOSEVAR -> SMALLCLOSE
35 DIVVAR -> DIV
36 DIVCLOSEVAR -> DIVCLOSE
37 AVAR -> A
38 ACLOSEVAR -> ACLOSE
39 BUTTONVAR -> BUTTON
40 BUTTONCLOSEVAR -> BUTTONCLOSE
41 FORMVAR -> FORM
42 FORMCLOSEVAR -> FORMCLOSE
43 TABLEVAR -> TABLE
44 TABLECLOSEVAR -> TABLECLOSE
45 TRVAR -> TR
46 TRCLOSEVAR -> TRCLOSE
47 TDVAR -> TD
48 TDCLOSEVAR -> TDCLOSE
```

CFG yang telah dibaca tadinya akan menghasilkan GNF. Pada file GNF.txt ini, akan dihasilkan PDA.

BAB V

KESIMPULAN

A. Kesimpulan

Dari pengerjaan tugas besar ini, dapat disimpulkan bahwa penggunaan PDA dapat digunakan sebagai HTML Checker untuk deteksi error pada sebuah program HTML.

B. Saran

Terdapat beberapa saran dari pengerjaan tugas besar ini agar ke depan nya, pengerjaan tugas-tugas besar lain dapat dilakukan dengan lebih baik dan terstruktur serta demi tidak mengulangi kesalahan yang sama pada tugas besar ini. Beberapa saran tersebut antara lain:

- Pengerjaan tugas besar ini terlalu mendekati deadline. Sebaiknya dalam tugas besar selanjutnya, para anggota kelompok harus lebih memperhatikan waktu pembuatan program serta laporan serta membuat timeline pengerjaan supaya seluruh tugas dalam kelompok dapat terselesaikan tepat waktu. Ditambah lagi para anggota kelompok yang jatuh sakit pada H-1 deadline sehingga menghambat pengerjaan tugas.
- Riset lebih lanjut mengenai tugas besar. Sebaiknya dalam pengerjaan tugas besar lain, para anggota kelompok lebih memahami materi dan melakukan riset terlebih dahulu terhadap materi utama yang menjadi pokok pembahasan dalam tugas besar tersebut.

C. Refleksi

Ada banyak hal yang bisa kita dapatkan ketika merefleksikan diri kita pada pengerjaan tugas besar ini. Perencanaan timeline menjadi sangat penting untuk dilakukan.

LAMPIRAN

Link ke repository github

<https://github.com/ibrahim-rasyid/TBTBFO>

Link ke diagram state

<https://app.diagrams.net/#G1vHiZd2cP1Md-vdd9gHuncdlpKLQVBizV>

Pembagian Tugas

Eduardus Alvito Kristiadi (13522004)	Pembuatan laporan, testing, debugging
Ibrahim Ihsan Rasyid (13522018)	PDA, parser, testing, debugging
Rici Trisna Putra (13522026)	Pembuatan parser, PDA, testing, debugging

DAFTAR PUSTAKA

Hopcroft, John E. et. al. 2006. *Introduction to Automata Theory, Languages, and Computation*

3rd Edition. Boston: Pearson Education, Inc.

<https://www.geeksforgeeks.org/what-is-context-free-grammar/>

<https://www.geeksforgeeks.org/introduction-of-pushdown-automata/>

W3School. "HTML Reference." *W3Schools* <https://www.w3schools.com/tags/default.asp>

LAMPIRAN

- Tautan Repository Github : <https://github.com/ibrahim-rasyid/TBTBFO>
- Tautan Desain Diagram State :
<https://drive.google.com/file/d/1vHiZd2cP1Md-vdd9gHuncdlpKLQVBizV/view?usp=sharing>
- Pembagian Tugas Kelompok :

NIM	Nama	Tugas
13522004	Eduardus Alvito Kristiadi	
13522018	Ibrahim Ihsan Rasyid	
13522026	Rici Trisna Putra	