Tugas Pemrograman
IF2124 Teori Bahasa Formal dan Otomata
HTML *Checker* dengan Pushdown Automata (PDA)

Oleh :

Maulana Muhammad Susetyo	13522127
Muhammad Dzaki Arta	13522149
Muhammad Rasheed Qais Tandjung	13522158

Kelompok: S → palingXanjay, X → duaharijadi

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

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, <img> 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, implementasikan sebuah program yang dapat memeriksa kebenaran HTML dari segi nama *tag* yang digunakan serta *attribute* yang dimilikinya. Pada tugas pemrograman ini, gunakanlah konsep Pushdown Automata (PDA) dalam mencapai hal tersebut yang diimplementasikan dalam bahasa **Python.**

Struktur HTML yang diharapkan (Perhatikan struktur dan urutan pada html, head, title, body).

```
<html>
<head>
<title> </title>
</head>
<body>
```

```
<h1>My First Heading</h1>
<p>My first paragraph.</p>
<!--… Elemen-elemen lain … -–>

</body>
</html>
```

Html selalu menjadi elemen terluar. Head selalu mendahului body. Title selalu di dalam head. Elemen-elemen lain berada di dalam body.

Mengingat banyaknya jenis *tags* beserta *attributes* yang tersedia pada HTML, *scope* tugas pemrograman akan dibatasi untuk memudahkan mahasiswa dalam pengerjaan. Batasan-batasan dari *tags* dan *attributes* yang diperiksa dapat dilihat di bawah ini.

Tag	Attribute Tambahan	Void Element	Catatan Tambahan
html			Tag html wajib ada, Dokumen harus diawali tag html
head			Tag head wajib ada, Tag head harus berada di dalam tag html dan diatas tag body
body			Tag body wajib ada, Tag body harus berada di dalam tag html dan dibawah tag head. Semua elemen yang disebutkan setelah ini kecuali title harus berada di dalam body, namun tidak semua tag wajib ada di dalam body.
title			hanya boleh berada dalam head
link	Rel, href	V	Atribut rel wajib ada, dapat berada dalam head
script	src		dapat berada dalam head
h1,h2, h3,h4, h5,h6			
p			
br		V	
em			

b			
abbr			
strong			
small			
hr		V	
div			
a	href		
img	src, alt	V	Atribut src wajib ada
button	type		Nilai attribute type dibatasi pada submit, reset, button
form	action, method		Nilai method harus dibatasi pada GET atau POST
input	type	V	Nilai attribute type harus dibatasi pada text, password, email, number, atau checkbox
table			Bentuk tabel tidak perlu diperhatikan (tidak apa-apa jika baris 1 terdiri dari 1 kolom sendangkan baris 2 terdiri dari 2 kolom)
tr			Tag tr harus berada dalam tag table
td			Tag td harus berada dalam tag table
th			Tag th harus berada dalam tag table

Catatan tambahan:

1. **Seluruh tag dapat memiliki *global attribute***. *Global attribute* yang akan digunakan pada tugas kali ini dibatasi pada atribut **id, class, dan style**.
2. Selain dari yang dibatasi pada Catatan Tambahan, nilai atribut selalu valid selama di dalam kutip dua (“). Misal pada atribut **id**, meskipun *whitespace* sebenarnya tidak diperbolehkan, pada scope tubes ini tidak diperhatikan.
3. Void Element merupakan elemen / singleton tag adalah elemen yang tidak memerlukan terdapat tag penutup untuk menjadi valid. Misal:

   <img src=”./img.png”>
4. Jangan lupa untuk **menghandle komentar** (<!-- ini komentar –->)

5. Terkait nesting pada tag, asisten hanya akan mengecek nesting pada:
    a. Html, head, body
    b. Tag div
    c. Html formatting element pada konten
    d. Tabel-tabelan

Bab II
LANDASAN TEORI

2.1 Definisi Push Down Automata

Pushdown automata merupakan jenis otomaton yang memperluas otomata berhingga non deterministik dengan transisi ε, dengan tambahan kemampuan *stack*. Ada dua varian pushdown automaton: satu menerima *input* dengan memasuki *final state*, yang lainnya dengan mengosongkan *stack*. Keduanya menerima tepatnya *context free language*. Secara informal, pushdown automata adalah otomata berhingga non deterministik yang dapat mengamati simbol *stack* untuk bertransisi. Dengan *stack*, pushdown automaton dapat "mengingat" informasi tak terbatas, tetapi hanya dapat mengaksesnya secara *last-in-first-out*. Contoh bahasa seperti palindrom genap dapat diakui oleh pushdown automaton. Meskipun mereka mengenali semua *context free language*, terdapat bahasa sederhana yang tidak dapat dikenali oleh pushdown automaton.

Notasi formal untuk Push Down Automation (PDA) melibatkan 7 komponen yaitu

$$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

Gambar 2.1 Rumus PDA

Keterangan :

Q: Sekumpulan keadaan yang terbatas, seperti keadaan *finite automata*.

Σ: Sekumpulan simbol masukan yang terbatas, juga analog dengan komponen yang bersesuaian dari *finite automata*.

Γ: *Stack* alphabet yang terbatas. Komponen ini, yang tidak memiliki *finite automata* analog, adalah kumpulan simbol yang boleh kita masukkan ke dalam *stack*.

δ: Fungsi transisi, δ mengambil 3 argumen δ(q, a, X). Dimana :

1. q adalah *state* di Q
2. a merupakan *input* simbol Σ atau a = ε, *empty* string, yang diasumsikan bukan merupakan simbol *input*.
3. X adalah simbol *stack*, yaitu anggota dari Γ.

Hasil dari δ adalah himpunan pasangan yang terbatas (p, ɣ). Dimana p adalah *state* baru dan ɣ adalah rangkaian simbol *stack* yang menggantikan X di bagian atas *stack*.

q_0: *Start state*. PDA berada dalam keadaan ini sebelum melakukan transisi apa pun.

Z_0: Start symbol. Awalnya, *stack* PDA terdiri dari satu contoh simbol ini, dan tidak ada yang lain.

F : Himpunan *state* bagian penerima, atau *final state*.

2.2 Bahasa dari PDA

Ada dua pendekatan dalam mendefinisikan bahasa yang diterima oleh Pushdown Automaton (PDA). Pertama, "penerimaan oleh *final state*," di mana PDA menerima *input* dengan mengonsumsinya dan mencapai *final state*. Kedua, "penerimaan oleh *empty stack*," di mana bahasa yang diterima adalah himpunan string yang menyebabkan PDA mengosongkan *stack*-nya, dimulai dari keadaan awal.

Meskipun kedua metode ini setara artinya, suatu bahasa L memiliki PDA yang menerimanya dengan keadaan akhir jika dan hanya jika L memiliki PDA yang menerimanya dengan *stack* kosong namun bahasa yang diterima oleh satu PDA dengan keadaan akhir dan *stack* kosong biasanya berbeda. Namun, kita dapat mengonversi PDA yang menerima bahasa dengan keadaan akhir menjadi PDA yang menerima dengan *stack* kosong, dan sebaliknya.

2.2.1 Diterima oleh *final state*

Misalkan $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ adalah sebuah PDA. Maka L(P), bahasa yang diterima oleh P berdasarkan *final state* adalah

$$\{w \mid (q_0, w, Z_0) \overset{*}{\underset{P}{\vdash}} (q, \epsilon, \alpha)\}$$

Gambar 2.2 Bahasa PDA yang diterima oleh *final state*

Jadi, untuk setiap keadaan akhir q dan setiap string *stack* a, PDA tersebut dapat mencapai keadaan akhir q tanpa memperhatikan isi *stack* pada saat itu. Dengan kata lain, PDA ini menerima bahasa dengan "*acceptance by final state*" karena, pada dasarnya, PDA hanya perlu mengonsumsi *input* hingga habis untuk mencapai keadaan akhir, dan kondisi *stack* saat itu tidak berpengaruh.

2.2.2 Diterima oleh *empty stack*

Untuk setiap PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F),$ kita juga mendefinisikan.

$$N(P) = \{w \mid (q_0, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \epsilon)\}$$

Gambar 2.3 Bahasa PDA yang diterima oleh *empty stack*

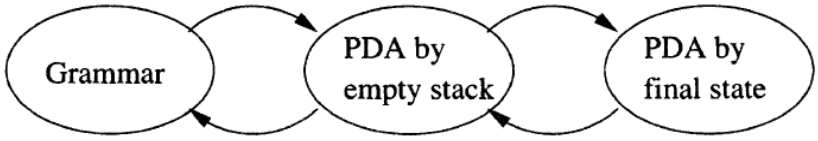
Jadi, untuk setiap keadaan q, himpunan N(P) mencakup semua *input* yang dapat dikonsumsi oleh PDA (P) dan pada saat yang sama mengosongkan *stack*-nya. Dengan kata lain, N(P) adalah himpunan *input* yang membuat PDA dapat mencapai keadaan akhir dengan *stack* kosong.

2.3 Kesetaraan antara PDA dan CFG

Sekarang, kita akan menunjukkan bahasa yang didefinisikan oleh PDA adalah *context free language*. Tujuannya adalah membuktikan bahwa tiga kelas bahasa berikut ini:

1. *Context free language*, yaitu bahasa yang didefinisikan oleh CFG.
2. Bahasa yang diterima oleh keadaan akhir oleh beberapa PDA.
3. Bahasa yang diterima oleh *stack* kosong oleh beberapa PDA.

Semuanya merupakan kelas yang sama. Kita sudah menunjukkan bahwa (2) dan (3) adalah sama. Ternyata langkah selanjutnya yang paling mudah adalah menunjukkan bahwa (1) dan (3) adalah sama, dengan demikian menyatakan kesetaraan dari ketiganya.



Gambar 2.4 Organisasi konstruksi yang menunjukkan kesetaraan tiga cara untuk mendefinisikan CFL's.

2.3.1 Dari *Grammar* ke PDA

Ideanya di balik konstruksi PDA dari sebuah *grammar* adalah agar PDA mensimulasikan urutan bentuk kalimat-kiri yang digunakan oleh *grammar* untuk menghasilkan string terminal tertentu w. Ekor setiap bentuk kalimat-kiri xAa muncul di *stack*, dengan A di bagian atas. Pada saat itu, x akan "diwakili" oleh kita yang telah mengonsumsi x dari *input*, meninggalkan apapun dari w yang mengikuti awalan x. Artinya, jika w = xy, maka y akan tetap ada di *input*.

Misalkan PDA berada dalam keadaan ID (q, y, Aa), mewakili bentuk kalimat-kiri xAa. PDA menebak produksi yang akan digunakan untuk mengembangkan A, katakanlah A → ß. Gerakan PDA adalah menggantikan A di bagian atas *stack* dengan ß, masuk ke ID (q, y, ßa). Perlu diperhatikan bahwa hanya ada satu keadaan, q, untuk PDA ini.

Sekarang (q, y, ßa) mungkin bukan representasi dari bentuk kalimat-kiri berikutnya, karena ß mungkin memiliki awalan terminal. Bahkan, ß mungkin tidak memiliki variabel sama sekali, dan a mungkin memiliki awalan terminal. Terminal apa pun yang muncul di awal ß perlu dihapus, untuk mengekspos variabel berikutnya di bagian atas *stack*. Terminal ini dibandingkan dengan simbol *input* berikutnya, untuk memastikan bahwa tebakan kita terhadap derivasi paling kiri dari string *input* w benar; jika tidak, cabang PDA ini mati.

Jika kita berhasil dengan cara ini untuk menebak derivasi paling kiri dari w, maka akhirnya kita akan mencapai bentuk kalimat-kiri ε. Pada saat itu, semua simbol di stack

entah sudah diperluas (jika mereka variabel) atau cocok dengan *input* (jika mereka terminal). Stack kosong, dan kita menerima dengan stack kosong.

Konstruksi informal di atas dapat dijelaskan secara lebih tepat sebagai berikut. Misalkan G = (V, T, Q, S) menjadi CFG. Konstruksikan PDA P yang menerima L(G) dengan kosong sebagai berikut:

$$P = (\{q\}, T, V \cup T, \delta, q, S)$$

Gambar 2.5 PDA P yang menerima L(G) dengan *stack* kosong

Dimana fungsi transisi δ didefinisikan sebagai:

1. Untuk setiap variabel A:

$$\delta(q, \epsilon, A) = \{(q, \beta) \mid A \rightarrow \beta \text{ is a production of } G\}$$

Gambar 2.6 Fungsi transisi untuk setiap variabel A

2. Untuk setiap terminal a:

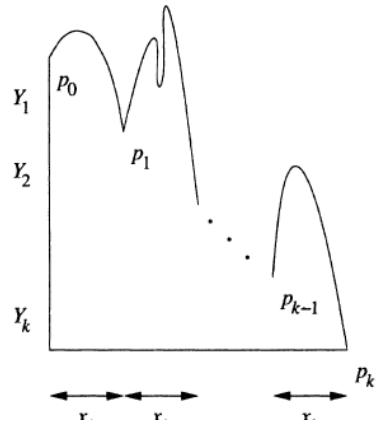
$$\delta(q, a, a) = \{(q, \epsilon)\}.$$

Gambar 2.7 Fungsi transisi untuk setiap terminal a

Teorema 1 : Jika PDA P dibangun dari CFG G melalui konstruksi di atas, maka N(P) = L(G).

2.3.2 Dari PDA ke *Grammar*

Sekarang, kita lengkapi bukti kesetaraan dengan menunjukkan bahwa untuk setiap PDA P, kita dapat menemukan CFG G yang bahasanya sama dengan bahasa yang diterima oleh P dengan stack kosong. Ide di balik bukti ini adalah menyadari bahwa peristiwa fundamental dalam sejarah pemrosesan *input* PDA adalah pencopotan bersih satu simbol dari stack, sambil mengkonsumsi beberapa *input*. PDA dapat mengubah keadaannya saat mengeluarkan simbol dari stack, jadi kita juga perlu mencatat keadaan yang masuk saat akhirnya ia mengeluarkan satu level dari stack-nya.



Gambar 2.8 PDA membuat serangkaian gerakan yang memiliki efek bersih memunculkan simbol dari *stack*

Gambar 2.8 menyarankan bagaimana kita mengeluarkan urutan simbol Y_1, Y_2, ..., Y□ dari stack. Sebuah *input* X_1 dibaca saat Y_1 dikeluarkan. Perlu dicatat bahwa "pop" ini adalah hasil bersih dari (mungkin) banyak langkah. Sebagai contoh, langkah pertama mungkin mengubah Y_1 menjadi simbol lain Z. Langkah berikutnya mungkin menggantikan Z dengan UV, langkah-langkah selanjutnya memiliki efek mengeluarkan U, dan kemudian langkah lain mengeluarkan V. Hasil bersihnya adalah Y_1 telah digantikan oleh kosong; yaitu, itu telah di-"pop", dan semua simbol *input* yang dikonsumsi sejauh ini membentuk X_1.

Kita juga menunjukkan dalam Gambar 6.10 perubahan bersih dari keadaan. Kita asumsikan bahwa PDA dimulai dalam keadaan P_0, dengan ? di bagian atas stack. Setelah semua langkah yang hasil bersihnya adalah mengeluarkan Y_1, PDA berada dalam keadaan P_1. Kemudian, ia melanjutkan untuk (hasil bersih) mengeluarkan ?, saat membaca string *input* X_2 dan mungkin, setelah banyak langkah, berakhir dalam keadaan P_2 dengan ? keluar dari stack. Perhitungan berlanjut hingga setiap simbol di stack dihapus.

Konstruksi kami dari tata bahasa yang setara menggunakan variabel yang masing-masing mewakili "peristiwa" yang terdiri dari:

1. Pengeluaran bersih dari beberapa simbol X dari stack, dan
2. Perubahan keadaan dari p pada awalnya menjadi q ketika X akhirnya telah digantikan oleh ε di stack.

Kami mewakili variabel tersebut dengan simbol gabungan (PXq). Ingat bahwa urutan karakter ini adalah cara kami untuk menggambarkan satu variabel; itu bukan lima simbol tata bahasa. Konstruksi formalnya diberikan oleh teorema berikut. Jika P = (Q, Σ, Γ, δ, q_0, Z_0,F). Maka ada sebuah CFG G dimana L(G)= N(P).

2.4 Deterministic Push Down Automata

Meskipun PDA, menurut definisinya, diizinkan untuk bersifat non deterministik, sub kasus yang deterministik sangat penting. Khususnya, parser pada umumnya berperilaku seperti PDA deterministik sehingga kelas bahasa yang dapat diterima oleh automata ini menarik untuk memberikan wawasan tentang konstruk apa yang cocok digunakan dalam bahasa pemrograman. Pada bagian ini, kita akan mendefinisikan PDA deterministik dan menyelidiki beberapa hal yang dapat dan tidak dapat dilakukan olehnya.

2.4.1 Defenisi dari Deterministic PDA

Secara intuitif, sebuah PDA dikatakan deterministik jika tidak pernah ada pilihan langkah dalam situasi apa pun. Pilihan ini terbagi menjadi dua jenis. Jika δ(q, a, X) mengandung lebih dari satu pasangan, maka PDA tersebut pasti nondeterministik karena kita dapat memilih di antara pasangan-pasangan tersebut saat memutuskan langkah selanjutnya. Namun, bahkan jika δ(q, a, X) selalu berupa himpunan singleton, kita masih bisa memiliki pilihan antara menggunakan simbol *input* nyata atau membuat langkah pada ε. Oleh karena itu, kita mendefinisikan PDA $P = (Q, \Sigma, \Gamma, q_0, Z_0, F)$ sebagai deterministik (sebuah PDA deterministik atau DPDA) jika dan hanya jika memenuhi kondisi-kondisi berikut:

1. δ(q, a, X) memiliki paling banyak satu anggota untuk setiap q di Q, a di Σ, atau a = ε, dan X di Γ.
2. Jika δ(q, ε, X) tidak kosong untuk suatu a di Σ, maka δ(q, ε, X) harus kosong.

2.4.2 Bahasa Regular dan Deterministik PDA

DPDA (Deterministic Pushdown Automata) menerima kelas bahasa yang berada di antara bahasa-bahasa reguler dan CFL (Context-Free Languages). Pertama, kita akan membuktikan bahwa bahasa-bahasa yang dapat diterima oleh DPDA mencakup semua bahasa reguler.

Teorema :

- Jika L adalah bahasa reguler, maka L = L(P) untuk suatu DPDA P.
- Sebuah bahasa L adalah N(P) untuk suatu DPDA P jika dan hanya jika L memiliki properti awalan dan L adalah L(P') untuk suatu DPDA P'.

2.4.3 DPDA dan CFG

DPDA dapat menerima bahasa seperti L_{wcwr} yang tidak bersifat reguler, seperti yang telah kita lihat sebelumnya. Untuk melihat bahwa bahasa ini tidak bersifat reguler, kita dapat menggunakan pumping lemma. Jika n adalah konstanta dari pumping lemma,

pertimbangkan string w = $0^n c 0^n$, yang termasuk dalam L_{wcwr}. Namun, ketika kita "pump" string ini, adalah grup pertama dari O yang panjangnya harus berubah, sehingga kita mendapatkan string dalam L_{wcwr} yang memiliki "penanda pusat" tidak berada di pusat. Karena string-string ini tidak ada dalam Lwcwr, kita mendapatkan kontradiksi dan menyimpulkan bahwa L_{wcwr} tidak bersifat reguler.

Di sisi lain, terdapat CFL seperti L_{wwr} yang tidak dapat diwakili oleh L(P) untuk DPDA P apa pun. Buktinya kompleks secara formal, tetapi intuisinya transparan. Jika P adalah DPDA yang menerima L_{wwr}, maka dengan urutan 0, P harus menyimpannya di stack atau melakukan sesuatu yang setara untuk menghitung jumlah O yang sewenang-wenang. Misalnya, ia bisa menyimpan satu X untuk setiap dua O yang dilihatnya, dan menggunakan keadaan untuk mengingat apakah jumlah tersebut genap atau ganjil.

Misalkan P telah melihat n O dan kemudian melihat 110^n. P harus memverifikasi bahwa ada n O setelah 11, dan untuk melakukannya, ia harus mengeluarkan elemen dari stack-nya. Sekarang, P telah melihat 0^n110^n. Jika ia melihat string yang identik selanjutnya, ia harus menerima, karena *input* lengkap berbentuk ww^R, dengan w =0^n110^n . Namun, jika ia melihat 0^n110^n untuk suatu m ≠ n, P tidak boleh menerima. Karena stack-nya kosong, ia tidak dapat mengingat integer n dengan benar, dan harus gagal mengenali L_{wwr} dengan benar. Kesimpulan kita adalah bahwa:

Bahasa yang diterima oleh DPDA melalui keadaan akhir dengan benar mencakup bahasa-bahasa reguler, tetapi secara tepat termasuk dalam CFL.

2.4.4 DPDA dan Grammar yang Ambigu

Kita dapat menyempurnakan kekuatan DPDA dengan mencatat bahwa bahasa-bahasa yang mereka terima semuanya memiliki tata bahasa yang tak ambigu. Namun sayangnya, bahasa-bahasa DPDA tidak persis sama dengan subset dari CFL yang tidak inheren ambigu. Sebagai contoh, L_{wwr} memiliki tata bahasa yang tak ambigu.

$$S \rightarrow 0S0 \mid 1S1 \mid \epsilon$$

Gambar 2.9 Grammar yang ambigu

Walaupun itu bukan merupakan bahasa DPDA. Teorema berikut menyempurnakan poin-poin diatas

Teorema :

- Jika L = N(P) untuk suatu DPDA P, maka L memiliki tata bahasa konteks-bebas yang tak ambigu.
- Jika L = L(P) untuk suatu DPDA P, maka L memiliki CFG yang tak ambigu.

Bab III
SPESIFIKASI TEKNIS PROGRAM

3.1 Main.py

Program utama yang memanggil fungsi-fungsi dari file lain dan menjalankan program untuk mengecek file html. Program ini menggunakan file Parser dan PDA.

3.2 Parser.py

Program parser adalah program untuk mengubah file PDA.txt menjadi PDA.py agar PDA dalam bentuk text tersebut dapat dijadikan sebuah kode yang dapat digunakan untuk mengecek validitas dari file html. Berikut adalah fungsi dan prosedur yang digunakan dalam program Parser.py.

3.2.1 ADVNEWLINE

Function ADVNEWLINE(file: TextIO) → **list of string**
I.S File PDA.txt sudah teridentifikasi
F.S Mengembalikan list of string dari 1 baris file PDA.

Fungsi ADVNEWLINE mengambil 1 baris dari file PDA.txt dan mengubah file tersebut menjadi list of string yang bisa digunakan dalam program python. Fungsi ini digunakan untuk membaca file PDA.txt.

3.2.2 FileToPDA

Function FileToPDA(filename: **String**) → PDA
I.S Menerima string filename
F.S Mengubah file filename.txt dan mengembalikan PDA

Program FileToPDA mengambil file PDA.txt dan mengembalikan PDA yang bisa digunakan dalam program main untuk mengecek file HTML. PDA merupakan class yang dibuat khusus untuk memudahkan program membaca fungsi PDA dari file txt.

3.2 PDA.py

File PDA.py berisikan class PDA serta fungsi processInput.

3.2.1 Class PDA

Merepresentasikan struktur data PDA yang berisi states, alphabet, stack symbols, fungsi transisi, startState, dan startSymbol. PDA juga memiliki komponen stack dari Stack.py, dan currentState yang menandakan state PDA pada saat tertentu.

Untuk pengerjaan bonus, Class PDA juga menyimpan data currentLine yang menandakan posisi line pembacaan PDA pada saat tertentu.

3.2.2 processInput

Fungsi processInput akan memanfaatkan struktur data PDA untuk mengecek apakah file HTML yang diinput merupakan kode yang valid. Kode PDA akan memproses input karakter per karakter dan menyesuaikannya dengan fungsi transisi yang dimiliki PDA. Jika PDA tidak menemukan fungsi transisi yang sesuai, program pengecekan akan berhenti dan processInput menghasilkan nilai *false*. Jika PDA melakukan iterasi sampai karakter input habis, maka processInput akan mengembalikan *true*.

3.3 Stack.py

Program Stack.py berisi class stack yang akan merepresentasikan stack pada PDA. Serta berisi fungsi/prosedur Top, Push, Pop yang merupakan perintah umum untuk stack.

3.3.1 Class Stack

Class Stack diinisialisasikan dengan menerima sebuah string sebagai simbol bottom untuk stack.

3.3.2 Top

Function Top() → **String**
I.S Sembarang
F.S Mengembalikan elemen teratas dari stack

Fungsi Top akan mengembalikan simbol paling atas pada stack.

3.3.3 Push

Procedure Push(word: **String**)

Prosedur Push berfungsi untuk memasukkan sebuah simbol ke dalam Stack sebagai simbol Top.

3.3.4 Pop

Procedure Pop()

Prosedur untuk mengambil elemen teratas dari stack. Setelah fungsi pop dipanggil maka elemen teratas dari stack akan hilang dari stack.

3.4 PDA.txt

PDA yang kami rancang merupakan *Deterministic PDA* bertipe *Accept by Empty Stack* yang melakukan pemrosesan input per karakter, dan akan selalu memiliki transisi yang mutlak untuk setiap kombinasi *state*, karakter, dan simbol tertentu. Terdapat beberapa karakter dan simbol-simbol penting yang kami gunakan untuk perancangan PDA:

- Simbol ***Z*** menandakan *Start Symbol*, dan sebagai acuan *stack* kosong dan input diterima.
- Simbol **W** menandakan *whitespace*/input spasi.
- Simbol **$ (dollar)** menandakan *epsilon*.
- Simbol **% (persen)** mendakan karakter *any*, yang merepresentasikan karakter apapun atau simbol stack apapun.
- Simbol # **(hash)** pada awal sebuah baris menandakan **komentar** dan memberikan perintah kepada *parser* untuk mengabaikan baris tersebut dan lanjut ke baris selanjutnya.

```
main html /html head /head
a b c d e f g h i j k l m n o p q r s t u v w x y z 1 2 3 4 5 6
< > ! - % / = W
a b c d e f g h i j k l m n o p q r s t u v w x y z 1 2 3 4 5 6
< > $ ! - % / = Z W C M B G A F T
main
Z
E
# ========================= State: html
=========================
main < Z main Z
main h Z html tml><head></head><body></body></html>Z
# Handling: Whitespace
main W Z main Z
# Handling: Comment
main ! Z main --CMBZ
main - - main $
main - C main $
main - M main $
```

```
main % C main C
main > B main $
# Handling: Reset Comment
main % M main CM
main % B main CMB
# ========================= State: html
=========================
html t t html $
html m m html $
html l l html $
html > > html $
# PDA Logic: Global Attributes
html W > html G>
# ID
html i G html d="A
html d d html $
html = = html $
html " " html $
html " A html $
html % A html A
# Class
html c G html lass="A
html c G html lass="A
html l l html $
html a a html $
html s s html $
# Style
html s G html tyle="A
html s G html tyle="A
html t t html $
html y y html $
html l l html $
html e e html $
# Handling: Comment
html ! h html --CMB<h
html - - html $
html - C html $
html - M html $
html % C html C
html > B html $
```

```
# Handling: Reset Comment
html % M html CM
html % B html CMB
# Transition: html --> head
html < < html $
html h h head $
# ========================= State: head
=========================
head < < head $
head h h head $
head e e head $
head a a head $
head d d head $
head > > head $
# PDA Logic: Global Attributes
head W > head G>
# ID
head i G head d="A
head d d head $
head = = head $
head " " head $
head " A head $
head % A head A
# Class
head c G head lass="A
head c G head lass="A
head l l head $
head a a head $
head s s head $
# Style
head s G head tyle="A
head s G head tyle="A
head t t head $
head y y head $
head l l head $
head e e head $
# Handling: Comment
head ! / head --CMB</
head - - head $
head - C head $
```

```
head - M head $
head % C head C
head > B head $
# Handling: Reset Comment
head % M head CM
head % B head CMB
# Transition: head --> title
head t / title itle></title></
# Transition: head --> link
head l / link inkW></
# Transition: head --> script
head s / script cript></script></
# Transition: head --> /head
head / / /head $
# ========================= State: link
=========================
link < < link $
link i i link $
link n n link $
link k k link $
# PDA Logic: Additional Attributes
# Rel
link W W link rel="A
link W W link rel="A
link r r link $
link e e link $
link l l link $
link = = link $
link " " link $
link " A link $
link " A link $
link % A link A
# PDA Logic: Optional Attributes
link W > link G>
# Alt
link h G link ref="A
link h G link ref="A
link r r link $
link e e link $
link f f link $
```

```
# PDA Logic: Global Attributes
# ID
link i G link d="A
link d d link $
link = = link $
link " " link $
link " A link $
link % A link A
# Class
link c G link lass="A
link c G link lass="A
link l l link $
link a a link $
link s s link $
# Style
link s G link tyle="A
link s G link tyle="A
link t t link $
link y y link $
link l l link $
link e e link $
# Transition: link --> head
link > > head $
# ========================= State: script
=========================
script < < script $
script s s script $
script c c script $
script r r script $
script i i script $
script p p script $
script t t script $
# PDA Logic: Optional Attributes
script W > script G>
# Src
script s G script rc="A
script s G script rc="A
script r r script $
script c c script $
script = = script $
```

```
script " " script $
script " A script $
script " A script $
script % A script A
# PDA Logic: Global Attributes
# ID
script i G script d="A
script d d script $
script = = script $
script " " script $
script " A script $
script % A script A
# Class
script c G script lass="A
script c G script lass="A
script l l script $
script a a script $
script s s script $
# Style
script s G script tyle="A
script s G script tyle="A
script t t script $
script y y script $
script l l script $
script e e script $
# Logic: <em>
script e / script m></em></
script m m script $
# Logic: <b>
script b / script ></b></
# Logic: <abbr>
script a / script bbr></abbr></
script b b script $
script r r script $
# Logic: <strong>
script s / script $
script t t script $
script r i script ong></strong></ti
script o o script $
script n n script $
```

```
script g g script $
# Logic: <small>
script m t script all></small></t
script a a script $
script l l script $
# Handling: Comment
script ! / script --CMB</
script - - script $
script - C script $
script - M script $
script % C script C
script > B script $
# Handling: Reset Comment
script % M script CM
script % B script CMB
# Transition: script --> /scriptTemp
script > > script $
script / / /scriptTemp $
# ========================= State: /scriptTemp
=========================
/scriptTemp s s /scriptTemp $
/scriptTemp c c /script $
/scriptTemp % % script $
# ========================= State: /script
=========================
/script < < /script $
/script / / /script $
/script s s /script $
/script c c /script $
/script r r /script $
/script i i /script $
/script p p /script $
/script t t /script $
# Transition: /script --> head
/script > > head $
# ========================= State: title
=========================
title < < title $
title t t title $
title i i title $
```

```
title t t title $
title l l title $
title e e title $
# PDA Logic: Global Attributes
title W > title G>
# ID
title i G title d="A
title d d title $
title = = title $
title " " title $
title " A title $
title % A title A
# Class
title c G title lass="A
title c G title lass="A
title l l title $
title a a title $
title s s title $
# Style
title s G title tyle="A
title s G title tyle="A
title t t title $
title y y title $
title l l title $
title e e title $
# Logic: <em>
title e / title m></em></
title m m title $
# Logic: <b>
title b / title ></b></
# Logic: <abbr>
title a / title bbr></abbr></
title b b title $
title r r title $
# Logic: <strong>
title s / title $
title t t title $
title r i title ong></strong></ti
title o o title $
title n n title $
```

```
title g g title $
# Logic: <small>
title m t title all></small></t
title a a title $
title l l title $
# Handling: Comment
title ! / title --CMB</
title - - title $
title - C title $
title - M title $
title % C title C
title > B title $
# Handling: Reset Comment
title % M title CM
title % B title CMB
# Transition: title --> /titleTemp
title > > title $
title / / /titleTemp $
# ========================= State: /titleTemp
=========================
/titleTemp t t /title $
/titleTemp % % title $
# ========================= State: /title
=========================
/title < < /title $
/title / / /title $
/title t t /title $
/title i i /title $
/title t t /title $
/title l l /title $
/title e e /title $
# Transition: /title --> head
/title > > head $
# ========================= State: /head
=========================
/head < < /head $
/head / / /head $
/head h h /head $
/head e e /head $
/head a a /head $
```

```
/head d d /head $
# Handling: Comment
/head ! / /head --CMB</
/head - - /head $
/head - C /head $
/head - M /head $
/head % C /head C
/head > B /head $
# Handling: Reset Comment
/head % M /head CM
/head % B /head CMB
# Transition: /head --> body
/head > > body $
# ========================= State: body
=========================
body < < body $
body b b body $
body o o body $
body d d body $
body y y body $
body > > body $
# Transition: body --> /bodyTemp
body < < body $
body / / /bodyTemp $
# PDA Logic: Global Attributes
body W > body G>
# ID
body i G body d="A
body d d body $
body = = body $
body " " body $
body " A body $
body % A body A
# Class
body c G body lass="A
body c G body lass="A
body l l body $
body a a body $
body s s body $
# Style
```

```
body s G body tyle="A
body s G body tyle="A
body t t body $
body y y body $
body l l body $
body e e body $
# PDA Logic: <p>
body p / body ></p></
# PDA Logic: <h1>, <h2>, ..., <h6>
body h / body h
body 1 h body ></h1></
body 1 1 body $
body 2 h body ></h2></
body 2 2 body $
body 3 h body ></h3></
body 3 3 body $
body 4 h body ></h4></
body 4 4 body $
body 5 h body ></h5></
body 5 5 body $
body 6 h body ></h6></
body 6 6 body $
# PDA Logic: <hr>
body r h body ></
# PDA Logic: <br>
body < < body $
body b / format F
# Transition: body --> a
body < < body $
body a / format A
# Transition: body --> img
body < < body $
body i / format I
# Transition: body --> form
body < < body $
body f / form orm></form></
# Transition: body --> input
body < < body $
body i / format I
# Transition: body --> table
```

```
body < < body $
body t / table able></table></
# Transition: body --> link
body < < body $
body l / linkBody inkW></
# PDA Logic: <div>
body < < body $
body d / body iv></div></
body i i body $
body v v body $
# PDA Logic: <b>
body b / format F
# PDA Logic: <abbr>
body a / format F
# PDA Logic: <em>
body < < body $
body e / body m></em></
body e e body $
body m m body $
# PDA Logic: <strong>
body < < body $
body s / format F
body r r body $
body o o body $
body n n body $
body g g body $
# PDA Logic: <small>
body s / format F
body a a body $
body l l body $
# PDA Logic: </button>
body b b body $
body u u body $
body t t body $
body o o body $
body n n body $
# Handling: Comment
body ! / body --CMB</
body ! b body --CMB<b
body - - body $
```

```
body - C body $
body - M body $
body % C body C
body > B body $
# Handling: Reset Comment
body % M body CM
body % B body CMB
# ========================= State: format
=========================
# Logic: <a>
format > A body </a></
format W A a G></a></
# Logic: <abbr>
format b A body br></abbr></
# Logic: <b>
format > F body </b></
# Logic: <br>
format r F body ></
# Logic: <button>
format u F button tton></button></
# Logic: <strong>
format t F body rong></strong></
# Logic: <small>
format m F body all></small></
# Logic: <script>
format c F scriptBody ript></script></
# Logic: <img>
format m I img gWsrc="A></
format m I img gWsrc="A></
# Logic: <input>
format n I input put></
# ========================= State: /bodyTemp
=========================
/bodyTemp b b /bodyTemp $
/bodyTemp o o /body $
/bodyTemp % % body $
# ========================= State: a =========================
a < < a $
a a a a $
# PDA Logic: Additional Attributes
```

```
a W > a G>
# Href
a h G a ref="A
a r r a $
a e e a $
a f f a $
a = = a $
a " " a $
a " A a $
# PDA Logic: Global Attributes
# ID
a i G a d="A
a d d a $
a = = a $
a " " a $
a " A a $
a % A a A
# Class
a c G a lass="A
a c G a lass="A
a l l a $
a a a a $
a s s a $
# Style
a s G a tyle="A
a s G a tyle="A
a t t a $
a y y a $
a l l a $
a e e a $
# Transition: a --> body
a > > body $
# ========================= State: /a =========================
# /a < < /a $
# /a / / /a $
# /a a a /a $
# /a > > body $
# ========================= State: img
=========================
img < < img $
```

```
img i i img $
img m m img $
img g g img $
# PDA Logic: Additional Attributes
# Src
img W W img $
img s s img $
img r r img $
img c c img $
img = = img $
img " " img $
img " A img $
img " A img $
img % A img A
# PDA Logic: Optional Attributes
img W > img G>
# Alt
img a G img lt="A
img a G img lt="A
img l l img $
img t t img $
# PDA Logic: Global Attributes
# ID
img i G img d="A
img d d img $
img = = img $
img " " img $
img " A img $
img % A img A
# Class
img c G img lass="A
img c G img lass="A
img l l img $
img a a img $
img s s img $
# Style
img s G img tyle="A
img s G img tyle="A
img t t img $
img y y img $
```

```
img l l img $
img e e img $
# Transition: img --> body
img > > body $
# ========================= State: button
=========================
button < < button $
button b b button $
button u u button $
button t t button $
button o o button $
button n n button $
# PDA Logic: Additional Attributes
button W > button G>
# Type
button t G button ype="T
button y y button $
button p p button $
button e e button $
button = = button $
button " " button $
# Type: submit
button s T button ubmit"
button u u button $
button b b button $
button m m button $
button i i button $
button t t button $
# Type: reset
button r T button eset"
button r T button eset"
button e e button $
button s s button $
button t t button $
# Type: button
button b T button utton"
button b T button utton"
button u u button $
button t t button $
button o o button $
```

```
button n n button $
# PDA Logic: Global Attributes
button W > button G>
# ID
button i G button d="A
button d d button $
button = = button $
button " " button $
button " A button $
button % A button A
# Class
button c G button lass="A
button c G button lass="A
button l l button $
button a a button $
button s s button $
# Style
button s G button tyle="A
button s G button tyle="A
button t t button $
button y y button $
button l l button $
button e e button $
# Transition: button --> body
button > > body $
# ========================= State: form
=========================
form < < form $
form f f form $
form o o form $
form r r form $
form m m form $
# PDA Logic: Additional Attributes
form W > form G>
# Action
form a G form ction="A
form c c form $
form t t form $
form i i form $
form o o form $
```

```
form n n form $
form = = form $
form " " form $
form " A form $
form % A form A
# Method
form m G form ethod="M
form e e form $
form t t form $
form h h form $
form o o form $
form d d form $
form = = form $
form " " form $
# Method: get
form g M form et"
form e e form $
form t t form $
# Method: post
form p M form ost"
form p M form ost"
form o o form $
form s s form $
form t t form $
# PDA Logic: Global Attributes
form W > form G>
# ID
form i G form d="A
form d d form $
form = = form $
form " " form $
form " A form $
form % A form A
# Class
form c G form lass="A
form c G form lass="A
form l l form $
form a a form $
form s s form $
# Style
```

```
form s G form tyle="A
form s G form tyle="A
form t t form $
form y y form $
form l l form $
form e e form $
# Transition: form --> /form
form > > body $
# ========================= State: /form
=========================
# /form < < /form $
# /form / / /form $
# /form f f /form $
# /form o o /form $
# /form r r /form $
# /form m m /form $
# /form > > body $
# ========================= State: input
=========================
input < < input $
input i i input $
input n n input $
input p p input $
input u u input $
input t t input $
# PDA Logic: Optional Attributes
input W > input G>
# Type
input t G input ype="T
input t G input ype="T
input y y input $
input p p input $
input e e input $
# Type: text
input t T input ext"
input t T input ext"
input e e input $
input x x input $
input t t input $
# Type: password
```

```
input p T input assword"
input p T input assword"
input a a input $
input s s input $
input w w input $
input o o input $
input r r input $
input d d input $
# Type: email
input e T input mail"
input e T input mail"
input m m input $
input a a input $
input i i input $
input l l input $
# Type: number
input n T input umber"
input n T input umber"
input u u input $
input m m input $
input b b input $
input e e input $
input r r input $
# Type: checkbox
input c T input heckbox"
input c T input heckbox"
input h h input $
input e e input $
input c c input $
input k k input $
input b b input $
input o o input $
input x x input $
# PDA Logic: Global Attributes
# ID
input i G input d="A
input d d input $
input = = input $
input " " input $
input " A input $
```

```
input % A input A
# Class
input c G input lass="A
input c G input lass="A
input l l input $
input a a input $
input s s input $
# Style
input s G input tyle="A
input s G input tyle="A
input t t input $
input y y input $
input l l input $
input e e input $
# Transition: input --> body
input > > body $
# ========================= State: table
=========================
table < < table $
table t t table $
table a a table $
table b b table $
table l l table $
table e e table $
# PDA Logic: Global Attributes
table W > table G>
# ID
table i G table d="A
table d d table $
table = = table $
table " " table $
table " A table $
table % A table A
# Class
table c G table lass="A
table c G table lass="A
table l l table $
table a a table $
table s s table $
# Style
```

```
table s G table tyle="A
table s G table tyle="A
table t t table $
table y y table $
table l l table $
table e e table $
# Transition: table --> /table
table > > /table $
# ========================= State: /table
=========================
/table < < /table $
/table / / /table $
/table t t /table $
/table a a /table $
/table b b /table $
/table l l /table $
/table e e /table $
/table > > body $
# Transition: /table --> tr
/table t / tr r></tr></
# Handling: Comment
/table ! / /table --CMB</
/table - - /table $
/table - C /table $
/table - M /table $
/table % C /table C
/table > B /table $
# Handling: Reset Comment
/table % M /table CM
/table % B /table CMB
# ========================= State: tr =========================
tr < < tr $
tr t t tr $
tr r r tr $
tr > > /tr $
# PDA Logic: Global Attributes
tr W > tr G>
# ID
tr i G tr d="A
tr d d tr $
```

```
tr = = tr $
tr " " tr $
tr " A tr $
tr % A tr A
# Class
tr c G tr lass="A
tr c G tr lass="A
tr l l tr $
tr a a tr $
tr s s tr $
# Style
tr s G tr tyle="A
tr s G tr tyle="A
tr t t tr $
tr y y tr $
tr l l tr $
tr e e tr $
# ========================= State: /tr
=========================
/tr < < /tr $
/tr / / /tr $
/tr t t /tr $
/tr r r /tr $
/tr > > /table $
# Transition: /tr --> td
/tr t / /tr /
/tr d / td ></td></
# Transition: /tr --> th
/tr h / th ></th></
# Handling: Comment
/tr ! / /tr --CMB</
/tr - - /tr $
/tr - C /tr $
/tr - M /tr $
/tr % C /tr C
/tr > B /tr $
# Handling: Reset Comment
/tr % M /tr CM
/tr % B /tr CMB
# ========================= State: td =========================
```

```
td < < td $
td t t td $
td d d td $
# PDA Logic: Global Attributes
td W > td G>
# ID
td i G td d="A
td d d td $
td = = td $
td " " td $
td " A td $
td % A td A
# Class
td c G td lass="A
td c G td lass="A
td l l td $
td a a td $
td s s td $
# Style
td s G td tyle="A
td s G td tyle="A
td t t td $
td y y td $
td l l td $
td e e td $
# Logic: <em>
td e / td m></em></
td m m td $
# Logic: <b>
td b / td ></b></
# Logic: <abbr>
td a / td bbr></abbr></
td b b td $
td r r td $
# Logic: <strong>
td s / td $
td t t td $
td r d td ong></strong></td
td o o td $
td n n td $
```

```
td g g td $
# Logic: <small>
td m t td all></small></t
td a a td $
td l l td $
# Handling: Comment
td ! / td --CMB</
td - - td $
td - C td $
td - M td $
td % C td C
td > B td $
# Handling: Reset Comment
td % M td CM
td % B td CMB
# Transition: td --> /td
td > > td $
td / / /tdTemp $
# ========================= State: /tdTemp
=========================
/tdTemp t t /td $
/tdTemp % % td $
# ========================= State: /td
=========================
/td < < /td $
/td / / /td $
/td t t /td $
/td d d /td $
/td > > /tr $
# ========================= State: th =========================
th < < th $
th t t th $
th h h th $
th > > th $
# PDA Logic: Global Attributes
th W > th G>
# ID
th i G th d="A
th d d th $
th = = th $
```

```
th " " th $
th " A th $
th % A th A
# Class
th c G th lass="A
th c G th lass="A
th l l th $
th a a th $
th s s th $
# Style
th s G th tyle="A
th s G th tyle="A
th t t th $
th y y th $
th l l th $
th e e th $
# Logic: <em>
th e / th m></em></
th m m th $
# Logic: <b>
th b / th ></b></
# Logic: <abbr>
th a / th bbr></abbr></
th b b th $
th r r th $
# Logic: <strong>
th s / th $
th t t th $
th r h th ong></strong></th
th o o th $
th n n th $
th g g th $
# Logic: <small>
th m t th all></small></t
th a a th $
th l l th $
# Handling: Comment
th ! / th --CMB</
th - - th $
th - C th $
```

```
th - M th $
th % C th C
th > B th $
# Handling: Reset Comment
th % M th CM
th % B th CMB
# Transition: th --> /th
th > > th $
th / / /thTemp $
# ========================= State: /thTemp
=========================
/thTemp t t /th $
/thTemp % % th $
# ========================= State: /th
=========================
/th < < /th $
/th / / /th $
/th t t /th $
/th h h /th $
/th > > /tr $
# ========================= State: linkBody
=========================
linkBody < < linkBody $
linkBody i i linkBody $
linkBody n n linkBody $
linkBody k k linkBody $
# PDA Logic: Additional Attributes
# Rel
linkBody W W linkBody rel="A
linkBody W W linkBody rel="A
linkBody r r linkBody $
linkBody e e linkBody $
linkBody l l linkBody $
linkBody = = linkBody $
linkBody " " linkBody $
linkBody " A linkBody $
linkBody " A linkBody $
linkBody % A linkBody A
# PDA Logic: Optional Attributes
linkBody W > linkBody G>
```

```
# Alt
linkBody h G linkBody ref="A
linkBody h G linkBody ref="A
linkBody r r linkBody $
linkBody e e linkBody $
linkBody f f linkBody $
# PDA Logic: Global Attributes
# ID
linkBody i G linkBody d="A
linkBody d d linkBody $
linkBody = = linkBody $
linkBody " " linkBody $
linkBody " A linkBody $
linkBody % A linkBody A
# Class
linkBody c G linkBody lass="A
linkBody c G linkBody lass="A
linkBody l l linkBody $
linkBody a a linkBody $
linkBody s s linkBody $
# Style
linkBody s G linkBody tyle="A
linkBody s G linkBody tyle="A
linkBody t t linkBody $
linkBody y y linkBody $
linkBody l l linkBody $
linkBody e e linkBody $
# Transition: linkBody --> body
linkBody > > body $
# ========================= State: scriptBody
=========================
scriptBody < < scriptBody $
scriptBody s s scriptBody $
scriptBody c c scriptBody $
scriptBody r r scriptBody $
scriptBody i i scriptBody $
scriptBody p p scriptBody $
scriptBody t t scriptBody $
# PDA Logic: Optional Attributes
scriptBody W > scriptBody G>
```

```
# Src
scriptBody s G scriptBody rc="A
scriptBody s G scriptBody rc="A
scriptBody r r scriptBody $
scriptBody c c scriptBody $
scriptBody = = scriptBody $
scriptBody " " scriptBody $
scriptBody " A scriptBody $
scriptBody " A scriptBody $
scriptBody % A scriptBody A
# PDA Logic: Global Attributes
# ID
scriptBody i G scriptBody d="A
scriptBody d d scriptBody $
scriptBody = = scriptBody $
scriptBody " " scriptBody $
scriptBody " A scriptBody $
scriptBody % A scriptBody A
# Class
scriptBody c G scriptBody lass="A
scriptBody c G scriptBody lass="A
scriptBody l l scriptBody $
scriptBody a a scriptBody $
scriptBody s s scriptBody $
# Style
scriptBody s G scriptBody tyle="A
scriptBody s G scriptBody tyle="A
scriptBody t t scriptBody $
scriptBody y y scriptBody $
scriptBody l l scriptBody $
scriptBody e e scriptBody $
# Logic: <em>
scriptBody e / scriptBody m></em></
scriptBody m m scriptBody $
# Logic: <b>
scriptBody b / scriptBody ></b></
# Logic: <abbr>
scriptBody a / scriptBody bbr></abbr></
scriptBody b b scriptBody $
scriptBody r r scriptBody $
```

```
# Logic: <strong>
scriptBody s / scriptBody $
scriptBody t t scriptBody $
scriptBody r i scriptBody ong></strong></ti
scriptBody o o scriptBody $
scriptBody n n scriptBody $
scriptBody g g scriptBody $
# Logic: <small>
scriptBody m t scriptBody all></small></t
scriptBody a a scriptBody $
scriptBody l l scriptBody $
# Handling: Comment
scriptBody ! / scriptBody --CMB</
scriptBody - - scriptBody $
scriptBody - C scriptBody $
scriptBody - M scriptBody $
scriptBody % C scriptBody C
scriptBody > B scriptBody $
# Handling: Reset Comment
scriptBody % M scriptBody CM
scriptBody % B scriptBody CMB
# Transition: scriptBody --> /scriptBody
scriptBody > > scriptBody $
scriptBody / / /scriptTempBody $
# ========================= State: /scriptTempBody
=========================
/scriptTempBody s s /scriptTempBody $
/scriptTempBody c c /scriptBody $
/scriptTempBody % % scriptBody $
# ========================= State: /scriptBody
=========================
/scriptBody < < /scriptBody $
/scriptBody / / /scriptBody $
/scriptBody s s /scriptBody $
/scriptBody c c /scriptBody $
/scriptBody r r /scriptBody $
/scriptBody i i /scriptBody $
/scriptBody p p /scriptBody $
/scriptBody t t /scriptBody $
# Transition: /scriptBody --> body
```

```
/scriptBody > > body $
# ========================= State: /body
=========================
/body b b /body $
/body o o /body $
/body d d /body $
/body y y /body $
# Handling: Comment
/body ! / /body --CMB</
/body - - /body $
/body - C /body $
/body - M /body $
/body % C /body C
/body > B /body $
# Handling: Reset Comment
/body % M /body CM
/body % B /body CMB
# Transition: /body --> /html
/body > > /body $
/body < < /body $
/body / / /html $
# ========================= State: /html
=========================
/html / / /html $
/html h h /html $
/html t t /html $
/html m m /html $
/html l l /html $
/html > > /html $
# Handling: Comment
/html < Z /html !--CMBZ
/html ! ! /html $
/html - - /html $
/html - C /html $
/html - M /html $
/html % C /html C
/html > B /html $
# Handling: Reset Comment
/html % M /html CM
/html % B /html CMB
```

BAB IV
PENGUJIAN DAN ANALISA

1. Uji Kasus 1

```
<html>
  <body>
    <h1>Hello, World!</h1>
    <p>This is a simple
webpage.</p>
  </body>
  <head>
    <title>Simple Webpage</title>
  </head>
</html>
```

```
$ python main.py
Reading debug.html...
Input:
<html>
  <body>
    <h1>Hello, World!</h1>
    <p>This is a simple webpage.</p>
  </body>
  <head>
    <title>Simple Webpage</title>
  </head>
</html>

------------------------------------------------
Rejected: Syntax Error

Invalid syntax at line 2:
<body>
------------------------------------------------
```

Pada kasus tersebut, <body> terletak setelah <head>, sehingga program mengembalikan **Rejected: Syntax Error**

2. Uji Kasus 2

```
<html>
   <head>
    <title>Simple Webpage</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <h2>Welcome to my page</h2>
    <img src=”./welcome.jpeg”>
    <p>This is a <em>simple</em>
webpage.</p>

     <div id=”footer” class=”footer”>
This is the end of the page </div>
  </body>
</html>
```

```
$ python main.py
Reading debug.html...
Input:
<html>
  <head>
    <title>Simple Webpage</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <h2>Welcome to my page</h2>
    <img src="./welcome.jpeg">
    <p>This is a <em>simple</em> webpage.</p>

    <div id="footer" class="footer"> This is the end
 of the page </div>
  </body>
</html>

------------------------------------------------
Accepted!
------------------------------------------------
```

Pada kasus ini, kode sudah memenuhi kriteria memiliki <html>, <head>, dan <body>, maka program mengembalikan **Accepted**.

3. Uji Kasus 3

```
<html>
  <head>
    <title>Simple Webpage</title>
  </head>
  <body>
    <!-- Bagian utama web -->
    <h1>Hello, World!</h1>
    <h2>Welcome to my page</h2>
    <hr>
    <img src="./welcome.jpeg"
alt="Welcome Banner">
    <p>This is a <em>simple</em>
webpage.</p>

    <!-- Custom element -->
    <div id=”footer” class=”footer”>
This is the end of the page </div>
  </body>
</html>
```

```
$ python main.py
Reading debug.html...
Input:
<html>
  <head>
    <title>Simple Webpage</title>
  </head>
  <body>
    <!-- Bagian utama web -->
    <h1>Hello, World!</h1>
    <h2>Welcome to my page</h2>
    <hr>
    <img src="./welcome.jpeg" alt="Welcome Banner">
    <p>This is a <em>simple</em> webpage.</p>


    <!-- Custom element -->
    <div id="footer" class="footer"> This is the end
 of the page </div>
  </body>
</html>


------------------------------------------------
Accepted!
------------------------------------------------
```

Program masih dapat melakukan pengecekan terhadap kode dengan komentar.

4. Uji Kasus 4

```
<html>
  <head>
    <title>Simple Webpage</title>
  </head>
  <body>
    <!-- Bagian utama web -->
    <h1>Hello, World!</h1>
    <h2>Welcome to my page</h2>
    <img alt="Welcome Banner">
    <p>This is a <em>simple</em>
webpage.</p>

    <!-- Custom element -->
    <div id="footer" class="footer">
This is the end of the page </div>
  </body>
</html>
```

```
$ python main.py
Reading debug.html...
Input:
<html>
  <head>
    <title>Simple Webpage</title>
  </head>
  <body>
    <!-- Bagian utama web -->
    <h1>Hello, World!</h1>
    <h2>Welcome to my page</h2>
    <img alt="Welcome Banner">
    <p>This is a <em>simple</em> webpage.</p>


    <!-- Custom element -->
    <div id="footer" class="footer"> This is the end
 of the page </div>
  </body>
</html>

------------------------------------------------
Rejected: Syntax Error

Invalid syntax at line 9:
<img alt="Welcome Banner">
------------------------------------------------
```

Pada kode tersebut, tag <img> tidak memiliki atribut src, sehingga program mengembalikan **Syntax Error**. Program juga dapat menuliskan nomor baris yang menghasilkan syntax error.

5. Uji Kasus 5

```
<html>
<head>
  <title>Simple Webpage</title>

</head>
<body>

<h2>HTML Forms</h2>

<form action="/action_page.php"
method="POST">
  <h5 class="label">First
name:</h5><br>
  <input type="text" id="fname"><br>
  <h5 class="label">Last
name:</h5><br>
  <input type="text"
id="lname"><br><br>
  <button
type="submit">Submit</button>
</form>

<p>If you click the "Submit" button,
the form-data will be sent to a page
called "/action_page.php".</p>

</body>
</html>
```

```
$ python main.py
Reading debug.html...
Input:
<html>
<head>
  <title>Simple Webpage</title>

</head>
<body>

<h2>HTML Forms</h2>

<form action="/action_page.php" method="POST">
  <h5 class="label">First name:</h5><br>
  <input type="text" id="fname"><br>
  <h5 class="label">Last name:</h5><br>
  <input type="text" id="lname"><br><br>
  <button type="submit">Submit</button>
</form>

<p>If you click the "Submit" button, the form-data will be sent to
a page called "/action_page.php".</p>

</body>
</html>


-----------------------------------------------
Accepted!
-----------------------------------------------
```

Program dapat melakukan pengecekan terhadap *tags* nested di dalam tag <form>.

6. Uji Kasus 6

```
<html>
<head>
  <title>Simple Webpage</title>

</head>
<body>

<h2>HTML Forms</h2>

<form action="/action_page.php"
method="POST">
  <h5 class="label">First
name:</h5><br>
  <input type="text" id="fname"><br>
  <h5 class="label">Last
name:</h5><br>
  <input type="text"
id="lname"><br><br>
  <button
type="submit">Submit</button>
</form>

<p>If you click the "Submit" button,
the form-data will be sent to a page
called "/action_page.php".</p>

</body>
</html>
```

```
$ python main.py
Reading debug.html...
Input:
<html>
<head>
  <title>Simple Webpage</title>

</head>
<body>

<h2>HTML Forms</h2>

<form action="/action_page.php" method="TEMBAK">
  <div id="label">First name:</div><br>
  <input type="text" id="fname"><br>
  <div id="label">Last name:</div><br>
  <input type="text" id="lname"><br><br>
  <button type="submit">Submit</button>
</form>

<p>If you click the "Submit" button, the form-data will be sent to
a page called "/action_page.php".</p>

</body>
</html>

------------------------------------------------
Rejected: Syntax Error

Invalid syntax at line 10:
<form action="/action_page.php" method="TEMBAK">
------------------------------------------------
```

Pada kasus di atas, program dapat mendeteksi bahwa atribut **method** pada tag <form> tidak sesuai (harusnya hanya berupa GET atau POST).

7. Uji Kasus 7

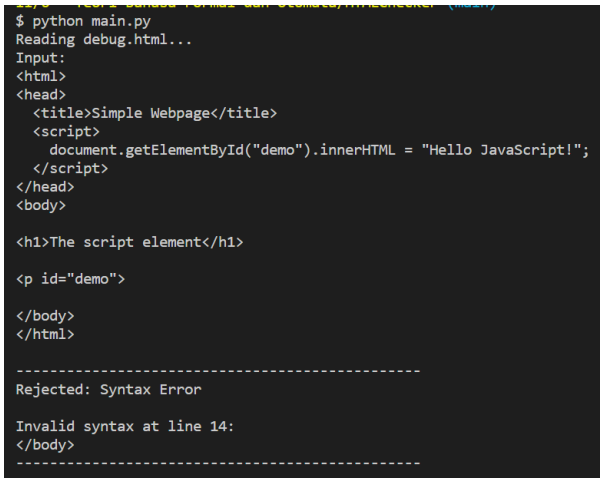
```
<html>
<head>
  <title>Simple Webpage</title>
  <script>

document.getElementById("demo").innerH
TML = "Hello JavaScript!";
  </script>
</head>
<body>

<h1>The script element</h1>

<p id="demo"></p>
```

```
$ python main.py
Reading debug.html...
Input:
<html>
<head>
  <title>Simple Webpage</title>
  <script>
    document.getElementById("demo").innerHTML = "Hello JavaScript!";
  </script>
</head>
<body>

<h1>The script element</h1>

<p id="demo"></p>

</body>
</html>

------------------------------------------------
Accepted!
------------------------------------------------
```

</body> </html>	

Program dapat melakukan pengecekan terhadap kode HTML dengan kode Javascript di dalam tag <script>.

8. Uji Kasus 8

<html> <head> <title>Simple Webpage</title> <script> document.getElementById("demo").innerHTML = "Hello JavaScript!"; </script> </head> <body> The script element <p id="demo"> </body> </html>	

Program mendeteksi bahwa tag <p> pada kode di atas tidak memiliki closing tag, dan mengembalikan **Rejected: Syntax Error.**

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari program yang kami buat kami dapat mengambil kesimpulan:

1. Keberhasilan Implementasi PDA:
   Program yang kami buat menggunakan konsep Pushdown Automata (PDA) mampu secara efektif menentukan validitas suatu file HTML. PDA memiliki kemampuan untuk melacak struktur dan hierarki tag dalam file HTML, sehingga memungkinkan deteksi kesalahan dengan baik.
2. Keterbatasan dan Batasan.
   Meskipun program ini dapat melakukan pengecekan validitas HTML, perlu diingat bahwa program kami memiliki batasan pada jenis tag dan atribut yang diperiksa. Pada implementasi ini, kami membatasi fokus pada beberapa tag dan atribut yang umum digunakan dalam struktur dasar HTML.
3. Kelebihan Validasi HTML
   Validasi HTML yang baik sangat penting untuk mendukung Search Engine Optimization (SEO), aksesibilitas, maintenance, kecepatan render, dan aspek profesionalisme suatu website. Program ini dapat membantu dalam memastikan bahwa struktur HTML terbentuk dengan baik dan sesuai standar.

5.2 Saran

Berdasarkan pengalaman pengerjaan program ini kami memberikan beberapa saran untuk perbaikan dan pengembangan selanjutnya.

1. Peningkatan kinerja terutama jika file HTML yang diperiksa berukuran besar.
2. Peningkatan efisiensi dalam penanganan string.
3. Perbaikan penanganan kesalahan dan penyajian pesan kesalahan yang lebih deskriptif
4. Integrasi dengan antarmuka pengguna grafis (GUI) dapat membuat program lebih user-friendly

DAFTAR PUSTAKA

John E. Hopcroft, Rajeev Motwani, Jeffrey D Ullman, "*Introduction To Automata Theory Languages, and Computation Third Edition,Pearson*, 2014".

LAMPIRAN

Github Repository: https://github.com/trimonuter/HTMLChecker
Diagram PDA:
https://drive.google.com/file/d/1u68E5FET1DR41bZSaoCgcJ2nOCGStXfX/view?usp=sharing

PEMBAGIAN TUGAS

Maulana Muhammad Susetyo (13522127) :

- Membuat diagram PDA
- Membantu pembuatan progam

Muhammad Dzaki Arta (13522149) :

- Membuat Laporan
- Membantu pembuatan diagram
- Membantu pembuatan progam

Muhammad Rasheed Qais Tandjung (13522158):

- Membantu pembuatan laporan
- Membantu pembuatan diagram
- Membuat program