

**MODUL PRAKTIKUM**  
**MODUL PRAKTIKUM TEORI BAHASA OTOMATA**

September 2016



Disusun oleh :  
Sri Handayaningsih  
Andri Pranolo

Laboratorium Basis Data  
Program Studi Teknik Informatika  
Fakultas Teknologi Industri  
Universitas Ahmad Dahlan

## Daftar Isi

Daftar Isi .....	i
Pertemuan I. Pengolahan String ..	1
Pertemuan II. Pengolahan String II .....	7
Pertemuan III. Operasi pada Bahasa .....	12
Pertemuan IV. Finite State Authomata (FSA) .....	18
Pertemuan V. Deterministic Finite Automata .....	24
Pertemuan VI. NonDeterministic Finite Automata .....	29
Pertemuan VII. NFA dengan transisi Epsilon .....	34
Pertemuan VIII. Chomsky Normal Form .....	39
Pertemuan IX. Penghilangan Rekursif Kiri .....	43
Pertemuan X. Normal Greibach .....	47

## Pengolahan String

Pertemuan	:	I
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	<ol style="list-style-type: none"><li>1. Mahasiswa mampu membuat rancangan <i>interface</i> untuk pengolahan string dengan menggunakan <i>visual programming</i></li><li>2. Mahasiswa mampu memahami algoritam dari pengolahan string dalam hal ini panjang string, <i>Reverse</i> dan <i>Concatenation</i></li></ol>
Indikator	:	<ol style="list-style-type: none"><li>1. Mahasiswa mampu membuat <i>interface</i> dengan menggunakan <i>visual programming</i></li><li>2. Mahasiswa mampu membuat fungsi pengolahan string untuk menghitung panjang string, <i>reverse</i>, dan <i>concatenation</i></li></ol>

### A. Dasar Teori

Terminologi dasar yang penting dalam memahami teori bahasa adalah alphabet, penyambungan (*Concatenation*) dan string pada alphabet  $V$ . alphabet digunakan untuk membentuk kata-kata di bahasa. Pada beberapa buku alphabet dilambangkan dengan  $\Sigma$ .

Kumpulan alphabet atau symbol disebut string. Ada banyak operasi pengolahan yang bisa dilakukan pada string yaitu *concatenation*, panjang string dan pembalikan (*reverse*).

1. *Concatenation* : penyambungan 2 karakter atau lebih membentuk suatu barisan karakter.
2. Panjang string : proses penghitungan jumlah karakter yang dimuat dalam suatu string
3. *Reverse* : pembalikan string

Misalnya  $u = abbbba$

$v = \text{bbbbba}$

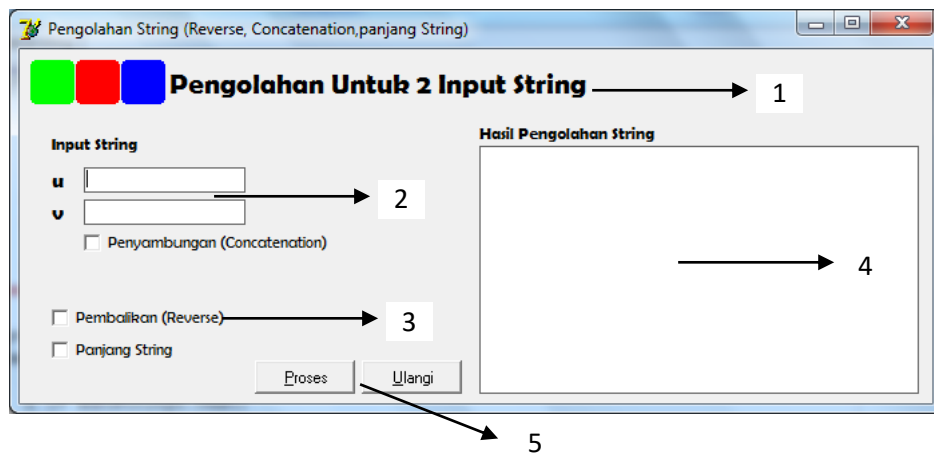
$uv = \text{abbbbabbbba}$  (penyambungan),

$|uv| = |u| + |v| = 6 + 6 = 12$  (panjang string)

$(uv)^R = (\text{abbbbabbbba})^R = \text{abbbbabbbba}$

## B. Langkah Praktikum

1. Buka program visual programming (dalam modul ini menggunakan Borland Delphi 7).
2. Buat Desain *form* seperti gambar 1.1



Gambar 1.1 . Desain *form* pengolahan string

No	Nama variable	Properties
1	<i>Label</i>	Posisi <i>Tab standard</i> <i>Caption</i> : diubah sesuai kebutuhan
2	<i>Edit</i>	Posisi <i>tab standard</i> <i>Name</i> : sesuaikan dengan input datanya
3	<i>Checkbox</i>	Posisi <i>tab standard</i> <i>Caption</i> : ubah sesuai dengan permintaan
4	<i>Memo</i>	Posisi <i>tab standard</i> <i>Name</i> : sesuaikan dengan kegunaan (missal memohasil) <i>Lines</i> : hapus semua teks yang ada di dalamnya
5	<i>Button</i>	Posisi <i>Tab standard</i> <i>Caption</i> : disesuaikan dengan kebutuhan <i>Name</i> : sesuai dengan nama <i>caption</i>

3. Jika desain *form* sudah selesai dilakukan masukkan coding di bawah ini :

#### Prosedur untuk menghitung panjang string

```
procedure TfrmUtama.Panjangstring;
var
  i: Integer;
begin
  u := edtstringu.Text;
  v := edtstringv.Text;

  if chkpenyambungan.Checked then
  begin
    mmohasil.Lines.Add('|uv| = |u| + |v| = '+IntToStr(Length(uv)))
  end
  else begin
    mmohasil.Lines.Add('|u| = '+IntToStr(Length(u)));
    mmohasil.Lines.Add('|v| = '+IntToStr(Length(v)));
  end;
end;
```

#### Prosedur untuk proses pembalikan pada string

```
procedure TfrmUtama.Pembalikan;
var
  panjangu, panjangv: Integer;
  i, j, total: Integer;
begin
  u := edtstringu.Text;
  v := edtstringv.Text;

  panjangv := length(v);
  panjangu := length(u);
  total := Length(uv);
  if chkpenyambungan.Checked then
  begin
    edttampung.Text := '';
    for i := 0 to Length(uv) do
    begin
      edttampung.Text := edttampung.Text+''+uv[total - i];
    end;
    mmohasil.Lines.Add('hasil pembalikan string penyambungan uv = '+
    edttampung.Text);
  end;
```

```

end
else begin
    edttampung.Text := '';
    for i:= 0 to Length(v) do
    begin
        edttampung.Text := edttampung.Text+v[panjangv - i];
    end;
    mmohasil.Lines.Add('hasil pembalikan string v = '+
edttampung.Text);

    edttampung.Text := '';
    for j:= 0 to Length(u) do
    begin
        edttampung.Text := edttampung.Text+'u[panjangu - j];
    end;
    mmohasil.Lines.Add('hasil pembalikan string u = '+
edttampung.Text);
    end;
end;

```


### Prosedur Penyambungan

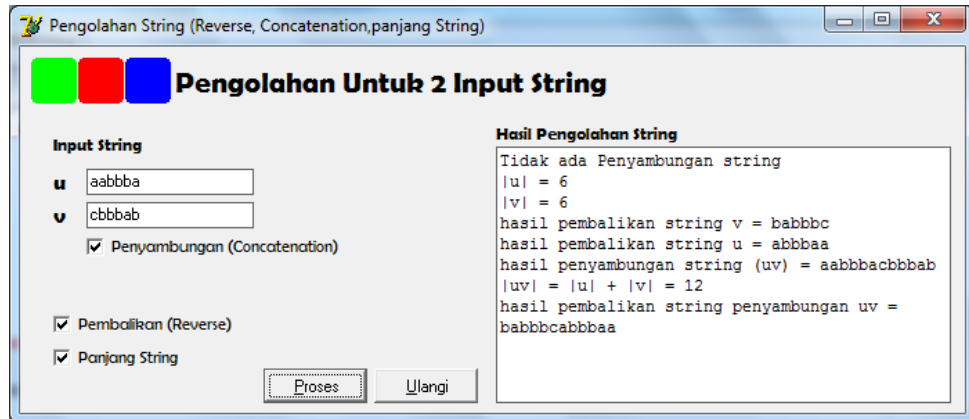
```

procedure TfrmUtama.Penyambungan;
begin
    u := edtstringu.Text;
    v := edtstringv.Text;

    if chkpenyambungan.Checked then
    begin
        uv := u+''+v;
        mmohasil.Lines.Add('hasil penyambungan string (uv) =
'+uv);
    end
    else begin
        mmohasil.Lines.Add('Tidak ada Penyambungan string')
    end;
end;

```

4. Tekan F9 atau tombol run  untuk menjalankan program
5. Coba masukkan 2 buah string dan lihat hasilnya cocokkan dengan hasil manualnya (contoh di bawah ini)



Gambar 1.2. *form pengolahan string dengan contoh output*

### C. Tugas Praktikum

Buat program untuk melakukan pembalikan dan untuk menghitung panjang 3 buah string dengan ketentuan sebagai berikut  $u = \text{aaaabbbbbb}$  ,  $v = \text{bbbccccca}$  ,  $w = \text{ccccbbbbaaa}$

<b>Nilai</b>	<b>Yogyakarta, .....</b>  <b>Paraf asisten</b>  <.....>
<b>Jawaban Postest</b>	



## Pengolahan String 2

Pertemuan	:	II
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	<ol style="list-style-type: none"><li>1. Mahasiswa mampu membuat rancangan <i>interface</i> untuk pengolahan string dengan menggunakan <i>visual programming</i></li><li>2. Mahasiswa mampu memahami algoritma dari pengolahan string dalam hal ini <i>Prefix</i>, <i>Suffix</i>, <i>Star Clouser</i> dan <i>Positif Clouser</i></li></ol>
Indikator	:	<ol style="list-style-type: none"><li>1. Mahasiswa mampu membuat <i>interface</i> dengan menggunakan <i>visual programming</i></li><li>2. Mahasiswa mampu membuat fungsi pengolahan string untuk <i>Prefix</i>, <i>Suffix</i>, <i>Star Clouser</i> dan <i>Positif Clouser</i></li></ol>

### A. Dasar Teori

Selain operasi penyambungan dan pembalikan ada beberapa operasi dasar pada string diantaranya *Prefix*, *suffix*, *star clouser* dan *positif clouser*.

*Prefix* string  $w$  adalah string yang dihasilkan dari string  $w$  dengan menghilangkan *nol* atau lebih simbol-simbol paling belakang dari string  $w$  tersebut.

Contoh :  $abc$ ,  $ab$ ,  $a$ , dan  $\epsilon$  adalah semua  $Prefix(x)$

*Postfix* (atau *Suffix*) string  $w$  adalah string yang dihasilkan dari string  $w$  dengan menghilangkan *nol* atau lebih simbol-simbol paling depan dari string  $w$  tersebut.

Contoh :  $abc$ ,  $bc$ ,  $c$ , dan  $\epsilon$  adalah semua  $Postfix(x)$ .

### Bahasa Universal

Jika  $\Sigma$  adalah alphabet, kita menggunakan  $\Sigma^*$  untuk menotasikan himpunan string(bahasa universal) yang dihasilkan oleh penggabungan *nol* atau lebih

symbol  $\Sigma$ .  $\Sigma^*$  selalu mengandung  $\lambda$  agar bisa mengeluarkan string yang kosong. Sedangkan  $\Sigma^+$  tidak mengandung string kosong  $\lambda$  atau  $\Sigma^+ = \Sigma^* - \lambda$ .

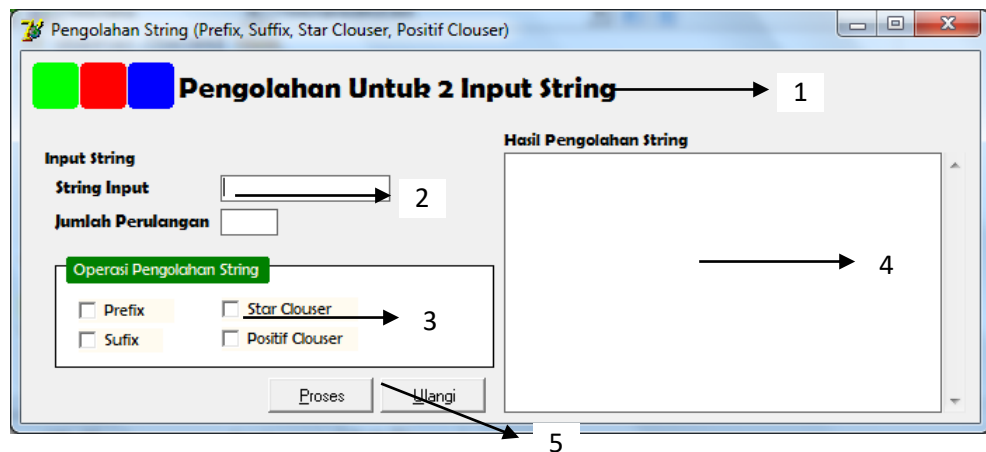
Contoh

$$\Sigma^* = \{ \lambda, a, aa, aaa, aaaa, \dots \}$$

$$\Sigma^+ = \{ a, aa, aaa, aaaa, \dots \}$$

## B. Petunjuk Praktikum

1. Buka program visual programming (dalam modul ini menggunakan Borland Delphi 7).
2. Buat Desain *form* seperti gambar 2.1



Gambar 2.1. Desain form pengolahan string

No	Nama variable	Properties
1	<i>Label</i>	Posisi <i>Tab standard</i> <i>Caption</i> : diubah sesuai kebutuhan
2	<i>Edit</i>	Posisi <i>tab standard</i> <i>Name</i> : sesuaikan dengan input datanya
3	<i>Checkbox</i>	Posisi <i>tab standard</i> <i>Caption</i> : ubah sesuai dengan permintaan
4	<i>Memo</i>	Posisi <i>tab standard</i> <i>Name</i> : sesuaikan dengan kegunaan (missal memohasil) <i>Lines</i> : hapus semua teks yang ada di dalamnya
5	<i>Button</i>	Posisi <i>Tab standard</i> <i>Caption</i> : disesuaikan dengan kebutuhan <i>Name</i> : sesuai dengan nama caption


3. Jika desain *form* sudah selesai dilakukan masukkan coding di bawah ini :

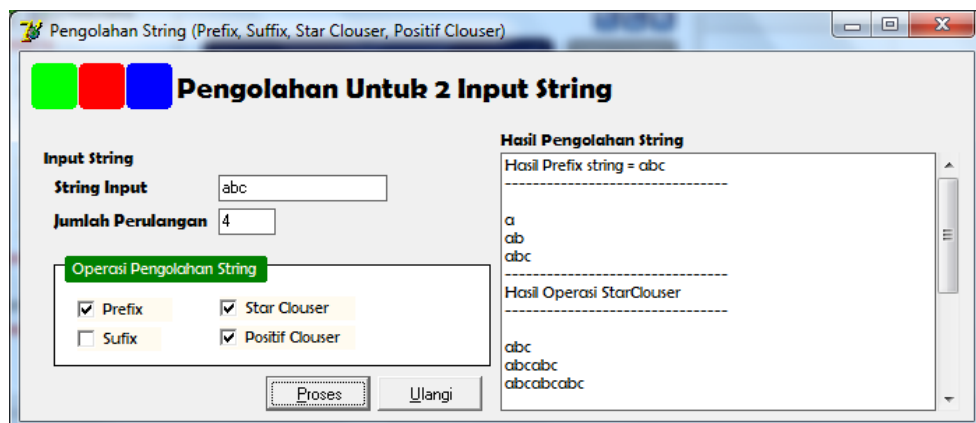
Prosedur untuk membuat *Prefix* dari string

```
procedure TfrmUtama.prefixmethod(karakter : string);
var
  i: Integer;
begin
  edttemplate.Text := '';
  mmohasil.Lines.Add('Hasil Prefix string = '+karakter);
  mmohasil.Lines.Add('-----');
  for i := 0 to Length(karakter) do
  begin
    edttemplate.Text :=edttemplate.Text+''+karakter[i];
    mmohasil.Lines.Add(edttemplate.Text)
  end;
  mmohasil.Lines.Add('-----');
end;
```

Prosedur membuat *starclouser* dari string

```
procedure TfrmUtama.starmethod(karakter : string;ulang : integer);
var
  i: Integer;
begin
  edttemplate.Text := '';
  mmohasil.Lines.Add('Hasil Operasi StarClouser ');
  mmohasil.Lines.Add('-----');
  mmohasil.Lines.Add('');
  for i := 0 to ulang-1 do
  begin
    edttemplate.Text := edttemplate.Text+''+karakter;
    mmohasil.Lines.Add(edttemplate.Text);
  end;
  mmohasil.Lines.Add('-----');
end;
```

4. Tekan F9 atau tombol run  untuk menjalankan program
5. Coba masukkan sebuah string dan lihat hasilnya cocokkan dengan hasil manualnya (gambar 2.2)



Gambar 2.2. form program ketika sudah di *run*

### **C. Tugas Praktikum**

Kembangkan program di atas, buat fungsi *Suffix* dan *Positif clouser*.

Tentukan hasilnya untuk string :

1. baababb
2. aaaaabbbab
3. babababbba
4. acaccaa

*(fungsi dan hasil dari pengolahan string di tuliskan di modul)*

<b>Nilai</b>	<b>Yogyakarta, .....</b>  <b>Paraf asisten</b>  <.....>
<b>Jawaban Postest</b>	

## Operasi Bahasa

Pertemuan	:	III
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	<ol style="list-style-type: none"> <li>1. Mahasiswa mampu membuat rancangan <i>interface</i> untuk pengolahan string dengan menggunakan <i>visual programming</i></li> <li>2. Mahasiswa mampu memahami algoritma dari operasi 2 buah bahasa. Dalam hal ini <i>Union</i>, <i>intersection</i>, <i>difference</i>, <i>concatenation</i>, <i>reverse</i></li> </ol>
Indikator	:	<ol style="list-style-type: none"> <li>1. Mahasiswa mampu membuat <i>interface</i> dengan menggunakan <i>visual programming</i></li> <li>2. Mahasiswa mampu membuat fungsi pengolahan bahasa <i>Union</i>, <i>intersection</i>, <i>difference</i>, <i>concatenation</i>, <i>reverse</i></li> </ol>

### A. Dasar Teori

Bahasa adalah subset dari  $\Sigma^*$

Contoh:  $\Sigma = \{a, b\}$

$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \dots\}$

Bahasa:  $\{\lambda\}$

$\{a, aa, aab\}$

$\{\lambda, abba, baba, aa, ab, aaaaaa\}$

Bahasa Tidak Terbatas  $L = \{a^n b^n : n \geq 0\}$

$\lambda$	}	$\in L$	$abb \notin L$
$ab$			
$aabb$			
$aaaaabbbbb$			

## Operasi dalam bahasa

Operasi-operasi pada bahasa tidak jauh berbeda dengan operasi-operasi pada string. Ada *Reverse*, *concatenation* dan lain sebagainya. Selain itu juga sama dengan operasi pada himpunan *Union*, *Intersection* dan *Difference*

Menggunakan operasi himpunan :  $\{a, ab, aaaa\} \cup \{bb, ab\} = \{a, ab, bb, aaaa\}$   
 $\{a, ab, aaaa\} \cap \{bb, ab\} = \{ab\}$   
 $\{a, ab, aaaa\} - \{bb, ab\} = \{a, aaaa\}$

Complement:  $\bar{L} = \Sigma^* - L$        $\overline{\{a, ba\}} = \{\lambda, b, aa, ab, bb, aaa, \dots\}$

## Reverse

Definisi:  $L^R = \{w^R : w \in L\}$

Contoh:  $\{ab, aab, baba\}^R = \{ba, baa, abab\}$

$$L = \{a^n b^n : n \geq 0\}$$

$$L^R = \{b^n a^n : n \geq 0\}$$

## Concatenation

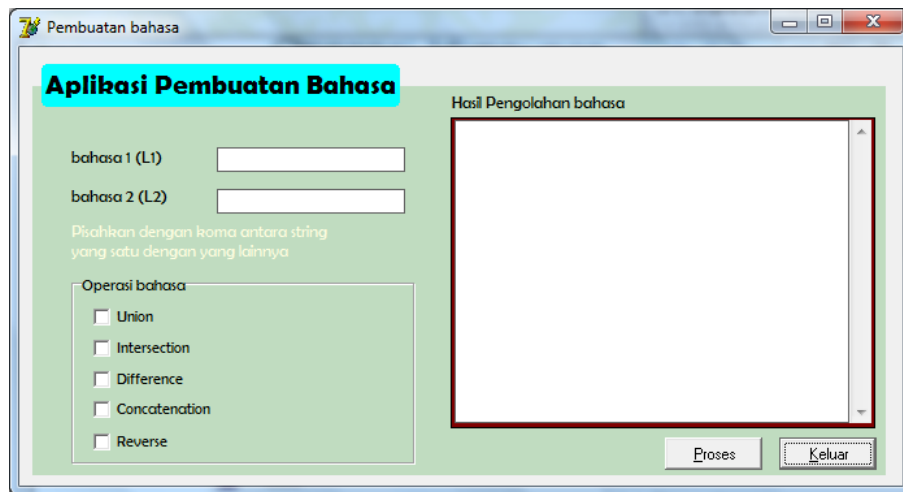
Definisi :  $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$

Contoh  $\{a, ab, ba\} \{b, aa\}$

$$= \{ab, aaa, abb, abaa, bab, baaa\}$$


## B. Petunjuk Praktikum

1. Buka program visual programming(dalam modul menggunakan Borland Delphi 7)
2. Desain form seperti di bawah ini (gambar 3.1)

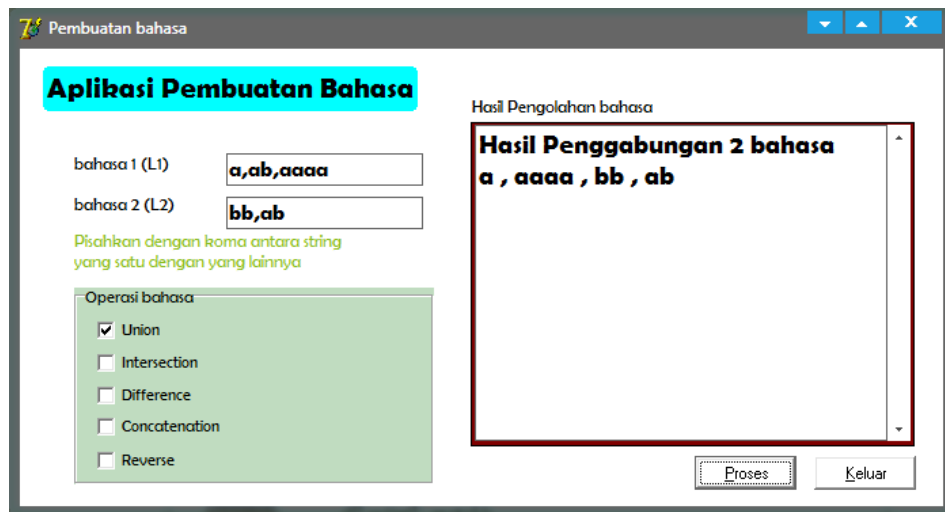


Gambar 3.1. desain form aplikasi pembuatan bahasa

No	Nama variable	Properties
1	<i>Label</i>	Posisi <i>Tab standard</i> <i>Caption</i> : diubah sesuai kebutuhan
2	<i>Edit</i>	Posisi <i>tab standard</i> <i>Name</i> : sesuaikan dengan input datanya
3	<i>Checkbox</i>	Posisi <i>tab standard</i> <i>Caption</i> : ubah sesuai dengan permintaan
4	<i>Memo</i>	Posisi <i>tab standard</i> <i>Name</i> : sesuaikan dengan kegunaan (missal memohasil) <i>Lines</i> : hapus semua teks yang ada di dalamnya
5	<i>Button</i>	Posisi <i>Tab standard</i> <i>Caption</i> : disesuaikan dengan kebutuhan <i>Name</i> : sesuai dengan nama <i>caption</i>

3. Buat fungsi untuk proses *union* dan *intersection*
4. Tekan F9 atau tombol run  untuk menjalankan program
5. Coba masukkan sebuah string dan lihat hasilnya cocokkan dengan hasil manualnya (contoh di bawah ini)





Gambar 3.2. contoh hasil pengolahan pada bahasa (operasi penggabungan)

Prosedur lengkap proses union (penggabugan 2 bahasa)

```

procedure Tfrmutama.memecahkanbahasa(kata : string);
var
  j: Integer;
  i: Integer;
  stringdata: string;
begin
  j := 0;
  for i := 1 to Length(kata)+1 do
  begin
    if (kata[i] <> ',') and (kata[i] <> '') then
    begin
      stringdata := stringdata+''+kata[i];
    end
    else begin
      lstdata.Items.Strings[j] := stringdata;
      Inc(j);
      stringdata:='';
    end;
  end;
end;
end;

```

```

procedure Tfrmutama.Union;
var
  k: Integer;
  j: Integer;
  hasilgabung: string;
  stringdata: string;
  i: Integer;
begin
  for i := lstdata.Items.Count - 1 downto 0 do
  begin
    if lstdata1.Items.IndexOf(lstdata.Items[i]) >= 0 then
      lstdata.Items.Delete(i);
    end;

    hasilgabung :=lstdata.Items.Strings[0];;
    mmohasil.Lines.Add('Hasil Penggabungan 2 bahasa');
    for j := 1 to lstdata.Items.Count - 1 do
    begin
      if lstdata.Items.Strings[j] <> '' then
        hasilgabung := hasilgabung+' , '+lstdata.Items.Strings[j];
      end;

      for k := 0 to lstdata1.Items.Count - 1 do
      begin
        hasilgabung := hasilgabung+' , '+lstdata1.Items.Strings[k];
      end;

      mmohasil.Lines.Add(hasilgabung);
      hasilgabung:=''
    end;
  end;
end;

```

### C. Tugas Praktikum

Dari program di atas tambahkan fungsi untuk proses *Concatenation*. Cek hasilnya untuk bahasa di bawah ini:

1.  $L1 = \{ab, bba\}$  dan  $L2 = \{bb, aba\}$
2.  $L1 = \{ac, bc\}$  dan  $L3 = \{bc, ab, ac\}$

<b>Nilai</b>	<b>Yogyakarta, .....</b>  <b>Paraf asisten</b>  <.....>
<b>Jawaban Postest</b>	

## ***Finite State Automata(FSA)***

Pertemuan	:	IV
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	<ol style="list-style-type: none"><li>1. Mahasiswa mampu membuat rancangan <i>interface</i> untuk <i>Finite State Automata</i></li><li>2. Mahasiswa mampu memahami algoritma <i>Finite State Automata</i></li></ol>
Indikator	:	<ol style="list-style-type: none"><li>1. Mahasiswa mampu membuat <i>interface</i> dengan menggunakan <i>visual programming</i></li><li>2. Mahasiswa mampu membuat fungsi dan aplikasi untuk <i>Finite State Automata</i></li></ol>

### **A. Dasar Teori**

*Finite automata* adalah mesin abstrak berupa sistem model matematika dengan masukan dan keluaran diskrit yang dapat mengenali bahasa paling sederhana (bahasa reguler) dan dapat diimplementasikan secara nyata dimana sistem dapat berada disalah satu dari sejumlah berhingga konfigurasi internal disebut state.

State sistem merupakan ringkasan informasi yang berkaitan dengan masukan-masukan sebelumnya yang diperlukan untuk menentukan perilaku sistem pada masukan-masukan berikutnya.

*Finite Automata* menggunakan prosedur yang saat diberikan masukan "string berhingga" akan berhenti

*Finite Automata* menyatakan "ya" dengan sejumlah berhingga komputasi jika string tersebut merupakan elemen bahasa sehingga lebih berfokus pada pengenalan dimana bila diberikan suatu program (string) akan menyatakan apakah string tersebut termasuk di bahasa atau tidak.

Secara formal *finite state automata* dinyatakan oleh 5 tupel atau  $M = (Q, \Sigma, \delta, S, F)$  dimana :

$Q$  = himpunan state/ kedudukan

$\Sigma$  = himpunan symbol *input*

$\delta$  = fungsi transisi

$S$  = state awal

$F$  = himpunan state akhir

Contoh : FSA untuk mengecek parity ganjil

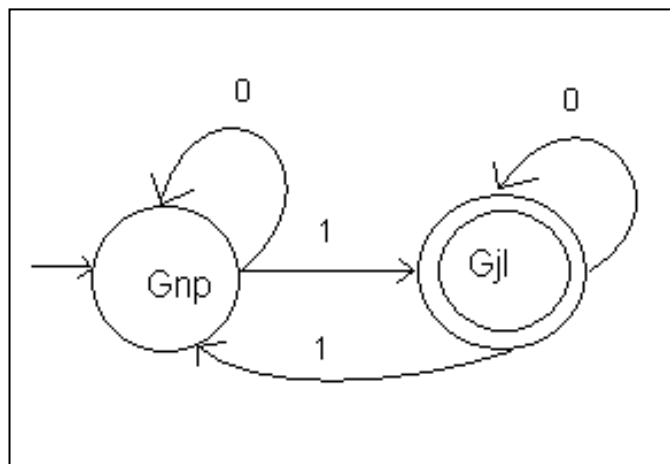
$Q = \{Gnp, Gjl\}$

diagram transisi

$\Sigma = \{0,1\}$

tabel transisi

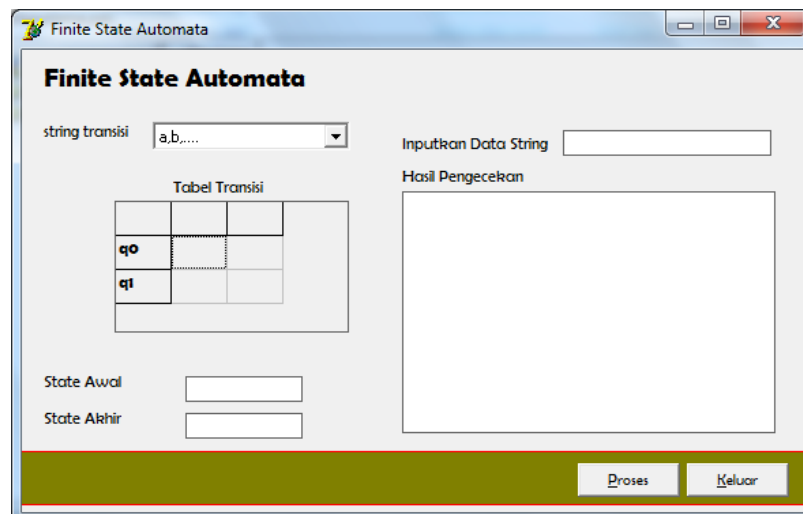
$\Delta$	0	1
Gnp	Gnp	Gjl
Gjl	Gjl	Gnp



$S = Gnp, F = \{Gjl\}$

## B. Petunjuk Praktikum

1. Jalankan visual programming
2. Buat desain form seperti di bawah ini



Gambar 4.1. Tampilan rancangan form FSA

### 3. Tambahkan potongan program di bawah ini

```
procedure Tfrmutama.FSAfungsi(state: string;transisi:string);
var
  i: Integer;
begin
  if (state='q0') and (transisi='a') then
  begin
    state:=strngrdtransisi.Cells[1,1];
  end;

  if (state='q0') and (transisi='b') then
  begin
    state:=strngrdtransisi.Cells[2,1];
  end;

  if (state='q1') and (transisi='a') then
  begin
    state:=strngrdtransisi.Cells[1,2];
  end;

  if (state='q1') and (transisi='b') then
  begin
    state:=strngrdtransisi.Cells[2,2];
  end;

  statehasil := state;
```

### **Tambahkan pada tombol proses**

```
procedure Tfrmutama.btnprosesClick(Sender: TObject);
var
  jumlahstring: Integer;
  i: Integer;
begin
  stateawal :=edtstateawal.Text;
  stateakhir := edtstateakhir.Text;
  jumlahstring := 0;
  for i := 1 to Length(Trim(edtinputdata.Text)) do
  begin
    FSAfungsi(stateawal,edtinputdata.Text[i]);

    mmketerangan.Lines.Add((''+stateawal+', ''+edtinputdata.Text[i]
    +'') = ''+statehasil);
    stateawal := statehasil;
    Inc(jumlahstring);
  end;

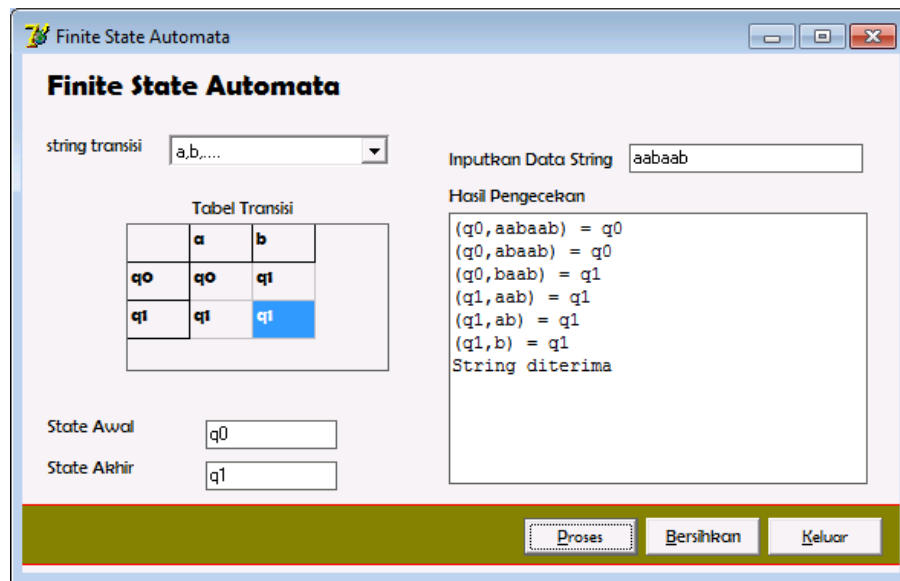
  if (statehasil =stateakhir) and
  (jumlahstring=Length(edtinputdata.Text)) then
```

```

begin
  mmoketerangan.Lines.Add('String diterima');
end
else begin
  mmoketerangan.Lines.Add('String ditolak');
end;
end;
end;

```

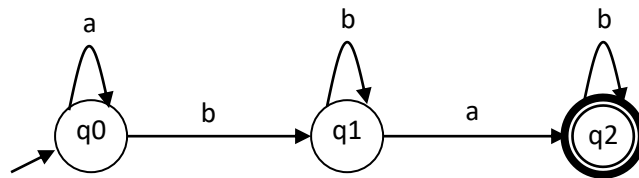
4. Jalankan program dan ujikan dengan string berikut ini string diterima :  
 abbab, string ditolak : abba



Gambar 4.2. Tampilan hasil running program FSA

### C. Tugas Praktikum

Tambahkan 1 *state* lagi dari program di atas, dengan q2 dan state q2 merupakan state akhir. Sehingga graf transisinya mnjadi sebagai berikut :



Tentukan apakah string berikut diterima atau di tolak

1. abbbba
2. bbbabb
3. babbb
4. bbbaaabab
5. babbababaa



<b>Nilai</b>	<b>Yogyakarta, .....</b>  <b>Paraf asisten</b>  <.....>
<b>Jawaban Postest</b>	

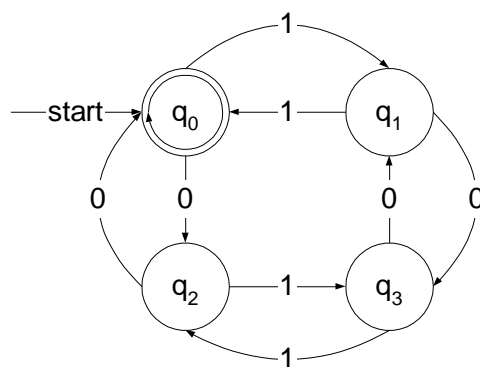
## *Deterministic Finite Automata (DFA)*

Pertemuan	:	V
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan <i>interface</i> untuk <i>Deterministic Finite Automata</i> 2. Mahasiswa mampu memahami algoritma <i>Deterministic Finite Automata (DFA)</i>
Indikator	:	1. Mahasiswa mampu membuat <i>interface</i> dengan menggunakan <i>visual programming</i> 2. Mahasiswa mampu membuat fungsi untuk mengecek DFA dan aplikasi DFA

### A. Dasar Teori

*Finite state automata* berdasarkan kemampuan berubah state-statenya bisa dikelompokkan ke dalam *deterministic* maupun *non-deterministic*.

DFA mempunyai cirri utama yaitu dari satu *state* tepat satu *state* berikutnya untuk setiap simbol masukan yang diterima. Lihat gambar di bawah



$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{0,1\}$$

$$S = q_0$$

$$F = \{q_0\}$$

fungsi transisi

$\delta$	0	1
q <sub>0</sub>	q <sub>2</sub>	q <sub>1</sub>
q <sub>1</sub>	q <sub>3</sub>	q <sub>0</sub>
q <sub>2</sub>	q <sub>0</sub>	q <sub>3</sub>
q <sub>3</sub>	q <sub>1</sub>	q <sub>2</sub>

Jika ada string

0011 : diterima.

10010 : ditolak, karena banyaknya 0 ganjil

$$\delta(q_0, 011) = \delta(q_2, 11) = \delta(q_3, 1) = q_2$$

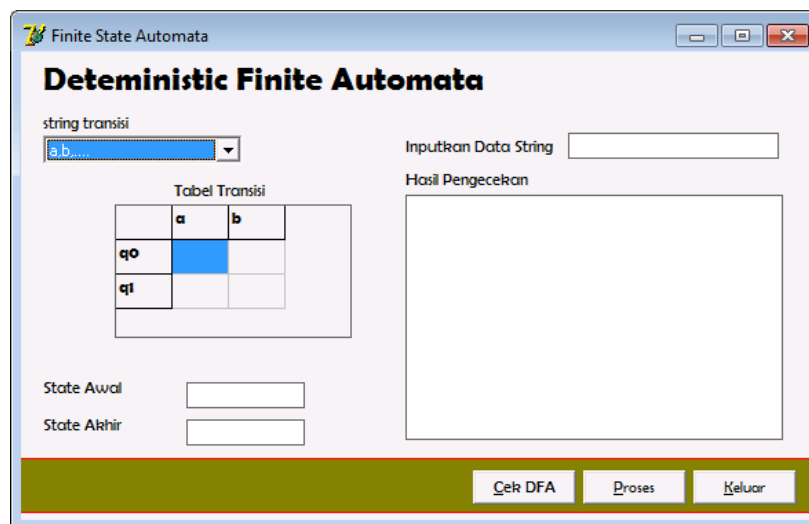
Ditolak

$$\delta(q_0, 1010) = \delta(q_1, 010) = \delta(q_3, 10) = \delta(q_2, 0) = q_0$$

Diterima

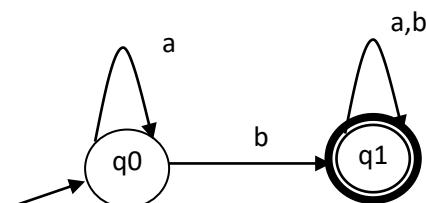
## B. Petunjuk Praktikum

1. Jalankan program *visual programming*
2. Buat program seperti pada *finite state automata*



Gambar 5.1. tampilan form *Deterministic Finite Automata*

3. Gunakan Diagram DFA di bawah ini untuk membuat program



Tabel Transisi

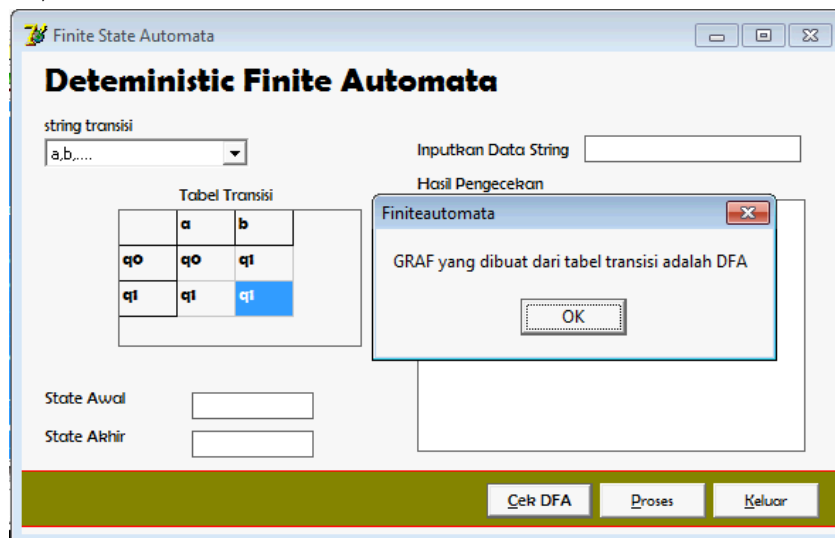
	a	b
q0	q0	q0
q1	q1	q1

4. Tambahkan fungsi untuk pengecekan bahwa graf DFA

```

procedure Tfrmutama.ccekDFA;
var
  j: Integer;
  i: Integer;
begin
  for j := 1 to strngrdtransisi.RowCount - 1 do
  begin
    if (strngrdtransisi.Cells[1,j] = '') and
    (strngrdtransisi.Cells[2,j] = '') then
    begin
      ShowMessage('bukan GRAF DFA, karena ada input yang kosong');
    end
    else begin
      ShowMessage('GRAF yang dibuat dari tabel transisi adalah
DFA');
    end;
  end;
end;
end;

```



Gambar 5.2. Tampilan form DFA cek graf

### C. Tugas Praktikum

Tambahkan 1 state dengan nama q2. Dan table transisinya

	A	B
q0	q0	q0
q1	q2	q1
q2	q2	q2

Gambar graf transisinya dan tentukan 2 string yang diterima dan ditolak (tuliskan hasilnya pada lembar jawaban di modul)

<b>Nilai</b>	<b>Yogyakarta, .....</b>  <b>Paraf asisten</b>  <.....>
<b>Jawaban Postest</b>	

## *Non-Deterministic Finite Automata (NFA)*

Pertemuan	:	VI
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	<ol style="list-style-type: none"><li>1. Mahasiswa mampu membuat rancangan <i>interface</i> untuk <i>Non-Deterministic Finite Automata</i></li><li>2. Mahasiswa mampu memahami algoritma <i>Non-Deterministic Finite Automata</i></li></ol>
Indikator	:	<ol style="list-style-type: none"><li>1. Mahasiswa mampu membuat <i>interface</i> dengan menggunakan <i>visual programming</i></li><li>2. Mahasiswa mampu membuat fungsi dan aplikasi untuk <i>Non-Deterministic Finite Automata</i></li></ol>

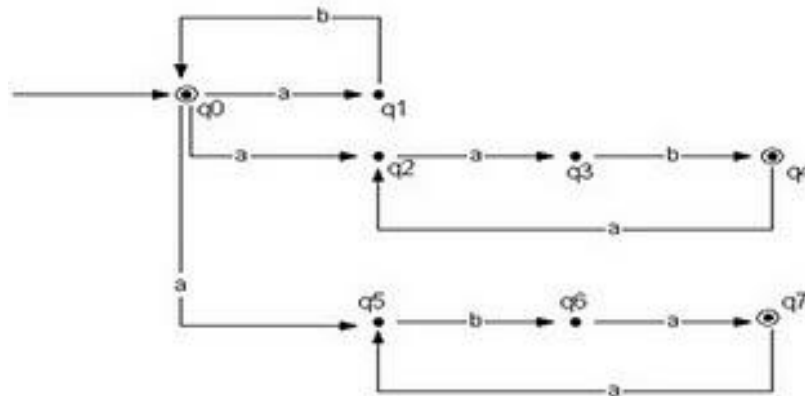
### **A. Dasar Teori**

*Nondeterministic Finite Automata (NFA)* adalah salah satu bagian dari otomata berhingga atau *Finite State Automata (FSA)*. Pada *Nondeterministic Finite Automata (NFA)* dimungkinkan satu simbol menimbulkan transisi ke lebih dari satu kondisi dan memberikan beberapa kemungkinan gerakan sehingga keluarannya tidak dapat dipastikan. Selain itu dimungkinkan juga terjadinya transisi spontan atau *transisi  $-\epsilon$* .

*Nondeterministic Finite Automata (NFA)* didefinisikan sebagai  $M$  yang merupakan sebuah koleksi dari 5 obyek ( $Q, \Sigma, s, F, \Delta$ ) dimana :

1.  $Q$  adalah sebuah himpunan hingga dari kedudukan-kedudukan.
2.  $\Sigma$  adalah sebuah abjad masukan.
3.  $s$  adalah salah satu kedudukan di dalam  $Q$  yang ditetapkan sebagai kedudukan permulaan.
4.  $F$  adalah sebuah koleksi dari kedudukan-kedudukan yang diterima atau final (koleksi / himpunan dari kondisi akhir).

5.  $\Delta$  adalah sebuah relasi pada  $(Q \times \Sigma) \times Q$  dan dinamakan relasi transisi. Salah satu rangkaian *Nondeterministic Finite Automata* (NFA) terlihat pada gambar.

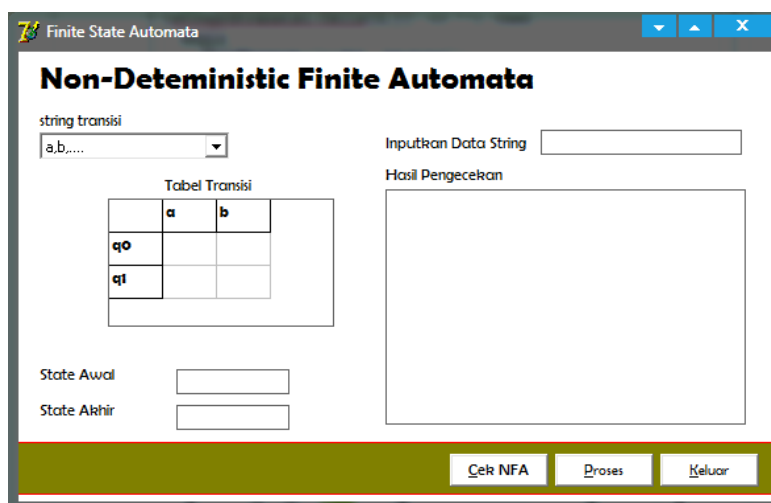


Gambar Rangkaian *Nondeterministic Finite Automata* (NFA)

Rangkaian pada Gambar tergolong dalam *Nondeterministic Finite Automata* (NFA) karena beberapa transisi yang berasal dari satu kondisi yaitu kondisi  $q_0$  memiliki inputan yang sama yaitu 'a'. Rangkaian tersebut akan menerima string ab, aab, aabaab, aba, dan abaaba, tetapi tidak akan menerima string abb dan aabb.

## B. Petunjuk Praktikum

1. Jalankan *visual programming*
2. Buat desain form seperti praktikum 3 dan 4 (lihat gambar di bawah ini).



Gambar 4.1 Tampilan form NFA



3. Tambahkan fungsi yang menandakan bahwa graf tersebut NFA dengan aturan

a. Dalam table transisi ada yang kosong maka dianggap sebagai graf NFA

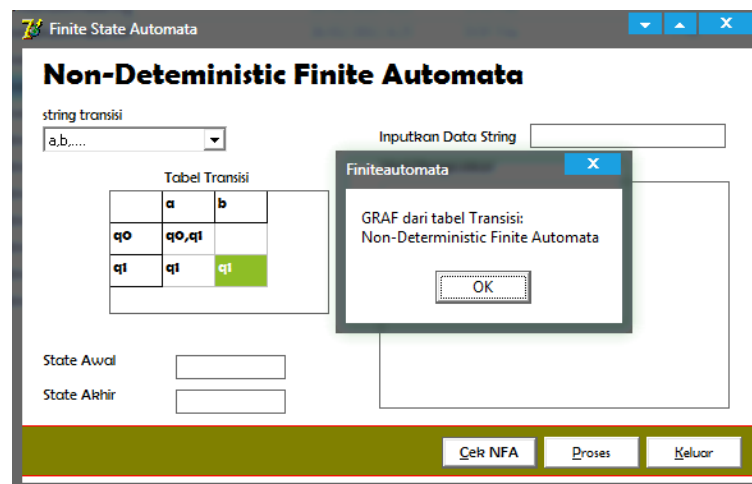
```

procedure Tfrmutama.cekNFA;
var
  j: Integer;
  i: Integer;
begin
  // TODO -cmm: Tfrmutama.cekDFA default body inserted
  {for i := 1 to strngrdtransisi.RowCount do
  begin
    InputStateA[i] := strngrdtransisi.Cells[1,i];
    InputStateB[i] := strngrdtransisi.Cells[2,i];
  end; }
  j:=1;

  if (strngrdtransisi.Cells[1,j] <> '') and
(strngrdtransisi.Cells[2,j] <> '') then
  begin
    ShowMessage('tidak ksosong');
    Inc(j);
  end
  else begin
    ShowMessage('GRAF dari tabel Transisi:'+#13+'Non-
Deterministic Finite Automata');
  end;
end;

```

b. Jika *input* transisi yang sama lebih dari 1 state maka di anggap sebagai NFA



Gambar 4.2 hasil pengecekan graf dari table traansisi

### **C. Tugas Praktikum**

Buat NFA yang menerima string di bawah ini : ab,aab,abb,aaab,abbb.  
Gunakan NFA dari kasus di atas dan kembangkan untuk mendapatkan string yang ditentukan. Gambar Graf transisinya.

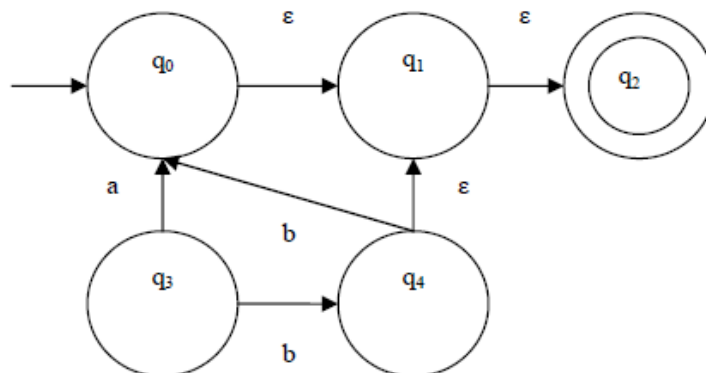
<b>Nilai</b>	<b>Yogyakarta, .....</b>  <b>Paraf asisten</b>  <.....>
<b>Jawaban Postest</b>	

## NFA dengan Transisi *Epsilon*( $\epsilon$ )

Pertemuan	:	VII
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan <i>interface</i> untuk NFA dengan transisi <i>epsilon</i> 2. Mahasiswa mampu memahami algoritma NFA dengan transisi <i>epsilon</i>
Indikator	:	1. Mahasiswa mampu membuat <i>interface</i> dengan menggunakan <i>visual programming</i> 2. Mahasiswa mampu membuat fungsi dan aplikasi untuk <i>deterministic Finite Automata</i>

### A. Dasar Teori

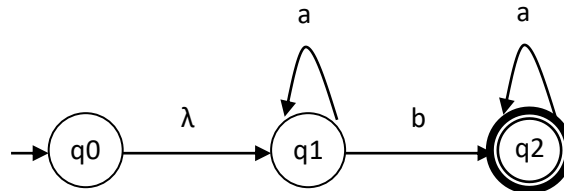
Pada NFA dengan transisi *epsilon*, diperbolehkan merubah *state* tanpa membaca *input*. Disebut dengan transisi *epsilon* karena tidak tergantung pada suatu *input* ketika melakukan transisi. Contoh :



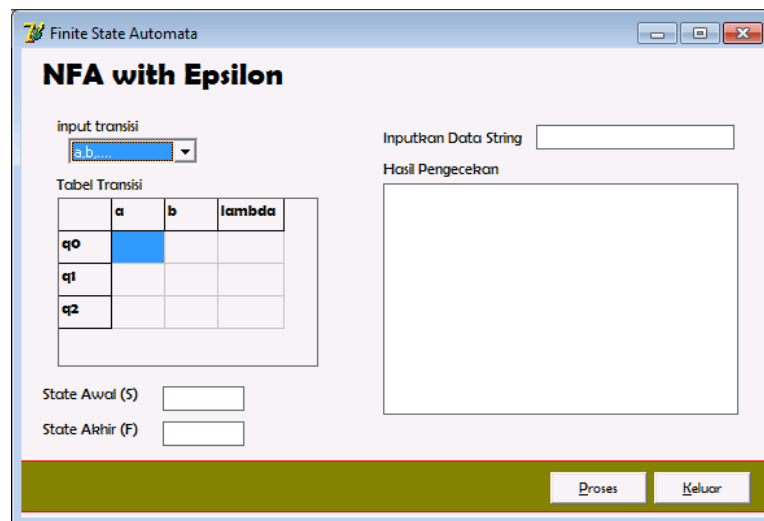
Contoh string yang diterima  $\epsilon$

## B. Petunjuk Praktikum

1. Buat Program seperti pada NFA (sesuaikan dengan graf di bawah).  
Tambah state dengan transisi *epsilon*, seperti gambar di bawah ini



2. Desain dengan menggunakan *visual programming* seperti gambar di bawah ini

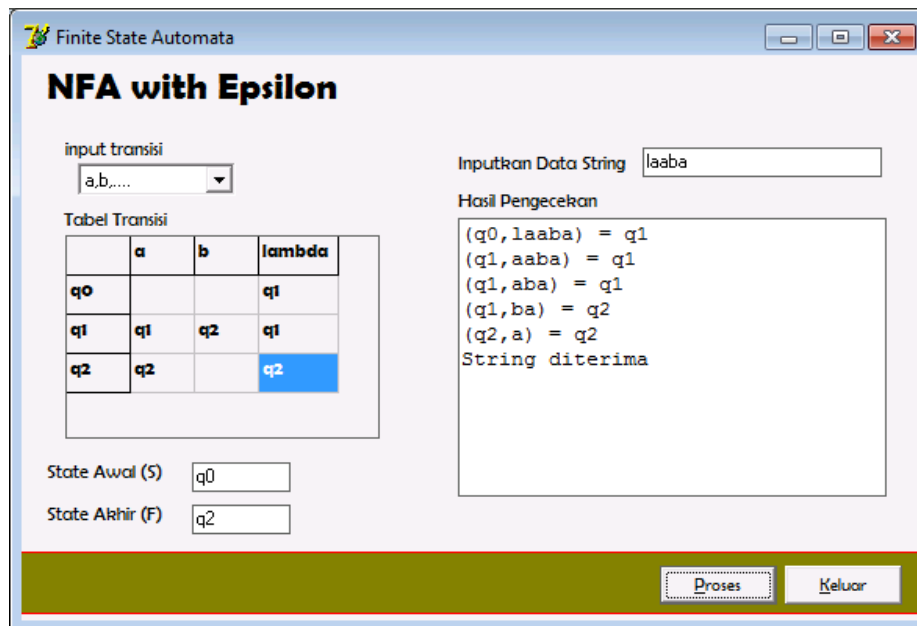


3. Tambahkan fungsi pada *form* yang sudah didesain. (yang perlu diingat bahwa NFA dan NFA- $\epsilon$  hampir sama, perbedaannya pada proses transisi. Pada NFA- $\epsilon$  transisi yang  $\epsilon$  bisa dilompati langsung.

Contoh Tabel Transisi untuk NFA- $\epsilon$

	a	b	E
q0	-	-	q1
q1	q1	q2	q1
q2	q2	-	q2

Untuk mengecek program lakukan validasi dengan memasukkan string di bawah ini. String yang di tolak :  $\lambda(1)$  dan string yang diterima : lab, laabab



### Algoritmanya

```

procedure NFAwithepsilon(statedata: string;transisi:string);
(mendefinisikan procedure untuk
var
  i: Integer;
begin
  if (statedata='q0') and (transisi='a') then
  begin
    state:=strngrdtransisi.Cells[1,1];
  end;

  if (statedata='q0') and (transisi='b') then
  begin
    state:=strngrdtransisi.Cells[2,1];
  end;

  if (statedata='q0') and (transisi='E') then
  begin
    state:=strngrdtransisi.Cells[3,1];
  end;

  if (statedata='q1') and (transisi='a') then
  begin
    state:=strngrdtransisi.Cells[1,2];
  end;

  if (statedata='q1') and (transisi='b') then
  begin
    state:=strngrdtransisi.Cells[2,2];
  end;

  if (statedata='q1') and (transisi='E') then
  begin
    state:=strngrdtransisi.Cells[3,2];
  end;

  if (statedata='q2') and (transisi='a') then
  begin

```

```

state:=strngrdtransisi.Cells[1,3];
end;

if (statedata='q2') and (transisi='b') then
begin
state:=strngrdtransisi.Cells[2,3];
end;

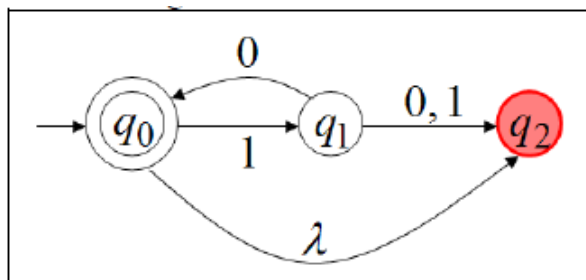
if (statedata='q2') and (transisi='E') then
begin
state:=strngrdtransisi.Cells[3,3];
end;

statehasil := state;
end;

```

### C. Tugas Praktikum

Buat program komputer dengan menggunakan visual programming untuk untuk menyelesaikan kasus di bawah ini:



Tentukan apakah string berikut diterima atau di tolak (ujikan dengan program yang anda buat)

1. 101
2. 10101010101010
3. 1101010

Gunakan program sebelumnya, kembangkan dari table transisinya

<b>Nilai</b>	<b>Yogyakarta, .....</b>  <b>Paraf asisten</b>  <.....>
<b>Jawaban Postest</b>	



## *Chomsky Normal Form (CNF)*

Pertemuan	:	VIII
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan <i>interface</i> CNF 2. Mahasiswa mampu memahami algoritama d dari proses <i>CNF</i>
Indikator	:	1. Mahasiswa mampu membuat <i>interface</i> dengan menggunakan <i>visual programming</i> 2. Mahasiswa mampu membuat fungsi untuk CNF

### **A. Dasar Teori**

Salah satu bentuk yang normal yang berguna untuk tata bahasa bebas konteks/context free grammar (CFG) adalah Bentuk Normal Chomsky/Chomsky Normal Form (CNF). CNF dari sebuah CFG dapat dibentuk setelah CFG disederhanakan, yaitu setelah CFG tidak memiliki produksi Useless, Unit, dan Nullable ( $\epsilon$ ). CFG telah dalam bentuk CNF apabila sisi kanan ( $\beta$ ) dari sebuah CNF berisi (Cole, 2007):

- Maksimal sebuah terminal, contoh  $A \rightarrow b$ , atau
- Maksimal terdiri dari dua buah variable  $A \rightarrow BC$ , atau
- Khusus pada aturan produksi symbol awal  $S \rightarrow \epsilon$ , jika  $\epsilon$  di dihasilkan oleh produksi S (dalam bahasa),
- Simbol awal S hanya muncul pada sisi kiri ( $\alpha$ ) aturan produksi.

Contoh bentuk normal Chomsky:

$$\begin{aligned} S &\rightarrow CD \\ C &\rightarrow AB \mid BD \\ A &\rightarrow a \\ B &\rightarrow b \\ D &\rightarrow d \end{aligned}$$

## B. Petunjuk Praktikum

1. Jalankan *visual programming* atau bahasa pemrograman lain yang dikuasai
2. Buat desain form input dan luaran untuk proses pembentukan normal Chomsky
3. Buat prosedur dan fungsi untuk membuat normal Chomsky berdasarkan algoritma sebagai berikut (Branicky, 2004):

1. Untuk setiap aturan produksi  $A \rightarrow B_1B_2 \dots B_k$  yang berisi sebuah terminal  $\sigma$ 
  - 1) Tambahkan symbol variable baru (non terminal) dan bentuk aturan produksi baru menjadi  $C\sigma \rightarrow \sigma$  (kecuali aturan produksi tersebut telah tersedia)
  - 2) Ganti setiap  $B_i = \sigma$  dengan  $C\sigma$
2. Ubah setiap aturan produksi  $A \rightarrow B_1B_2 \dots B_k$  dengan  $B_1 \dots B_k$  berupa symbol variable dan  $k \geq 3$ , maka lakukan
  - 1)  $A \rightarrow B_1A_1$
  - 2)  $A_1 \rightarrow B_2A_2$
  - 3)  $A_2 \rightarrow B_3A_3$
  - 4) ...
  - 5)  $A_{k-2} \rightarrow B_{k-1}B_k$  dimana  $A_i$  adalah variable baru
3. Gabungkan hasil pada langkah 1 dan 2

4. Lakukan testing untuk memastikan program berjalan dengan benar
5. Dokumentasikan form/interface, prosedur, fungsi, input dan luaran program yang dibuat dalam kertas A4, diketik dengan huruf times new roman 12 dengan 1 spasi.

### C. Pretest Praktikum

Buatlah bentuk CNF dari CFG sebagai berikut menggunakan program yang telah anda buat.

$$S \rightarrow AB \mid Ba$$

$$A \rightarrow dB \mid b$$

$$B \rightarrow Bb \mid a$$

$$C \rightarrow c$$

### D. Postest Praktikum

Gunakan program yang anda buat untuk membuat CNF dari CFG berikut:

$$S \rightarrow ABCD \mid BCd \mid CD$$

$$A \rightarrow bB \mid Cd$$

$$B \rightarrow cD \mid b$$

$$C \rightarrow c$$

$$D \rightarrow aBc \mid d$$

### E. Referensi

Branicky, M. (2004). EECS 343 Handout: Chomsky Normal Form. Retrieved from <http://dora.eeap.cwru.edu/msb/343/normal.pdf>

Cole, R. (2007). *Converting CFGs to CNF (Chomsky Normal Form)*. Retrieved from <http://www.cs.nyu.edu/courses/fall07/V22.0453-001/cnf.pdf>

<b>Nilai</b>	<b>Yogyakarta, .....</b>  <b>Paraf asisten</b>  <.....>
--------------	---

**Jawaban Posttest**

## *Penghilangan Rekursif Kiri*

Pertemuan	:	IX
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan <i>interface</i> penghilangan rekursif kiri 2. Mahasiswa mampu memahami algoritama d dari proses <i>penghilangan rekursif kiri</i>
Indikator	:	1. Mahasiswa mampu membuat <i>interface</i> dengan menggunakan <i>visual programming</i> atau <i>bahasa pemrograman lain yang dikuasai</i> 2. Mahasiswa mampu membuat prosedur/fungsi untuk penghilangan rekursif kiri

### **A. Dasar Teori**

Conext Free Grammar (CFG) dikatakan rekursif kiri apabila aturan produksinya dalam bentuk (Maza, 2004):

$$A \rightarrow A\beta$$

Dimana A merupakan symbol variable (non terminal) dan  $\beta$  adalah string dari symbol-simbol CFG. Rekursif kiri akan mengakibatkan perulangan penurunan CFG yang tidak akan pernah berhenti (Weimar et al., 2015). Oleh karena itu, rekursif kiri dalam CFG harus dihilangkan.

Contoh aturan produksi rekursif kiri:

$$S \rightarrow SdBa$$

$$A \rightarrow AbCD$$

$$C \rightarrow Cda$$

$$D \rightarrow DbacBa$$

Langkah-langkah untuk penghilangan rekursif kiri (Moore, 2001; Utdirartamo, 2005):

1. Pisahkan aturan produksi yang rekursif kiri dan yang tidak, missal:
  - a. Aturan produksi yang rekursif kiri  $A \rightarrow A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \mid \dots \mid A\alpha_n$ , sehingga dapat diketahui  $\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n$
  - b. Aturan produksi yang tidak rekursif kiri  $A \rightarrow \beta_1 \mid \beta_2 \mid \beta_3 \mid \dots \mid \beta_m$ , sehingga dapat diketahui  $\beta_1, \beta_2, \beta_3, \dots, \beta_m$

- Lakukan penggantian aturan produksi yang terdapat rekursif kiri berdasarkan rumus sebagai berikut:

$$A \rightarrow \beta_1 \mid \beta_1 A' \mid \beta_2 \mid \beta_2 A' \mid \beta_3 \mid \beta_3 A' \mid \dots \mid \beta_m \mid \beta_m A'$$

$$A' \rightarrow \alpha_1 \mid \alpha_1 A' \mid \alpha_2 \mid \alpha_2 A' \mid \alpha_3 \mid \alpha_3 A' \mid \dots \mid \alpha_n \mid \alpha_n A'$$

Dimana  $A'$  merupakan symbol variable (non terminal) yang tidak digunakan di aturan produksi lain dalam CFG tersebut.

## B. Petunjuk Praktikum

- Jalankan *visual programming* atau bahasa pemrograman lain yang dikuasai
- Buat desain form input dan luaran untuk proses penghilangan rekursif kiri.
- Buat prosedur/fungsi untuk proses penghilangan rekursif kiri berdasarkan algoritma sebagai berikut:

- Pisahkan produksi yang rekursif kiri dan tidak
  - Produksi rekursif kiri  $A\alpha_1, A\alpha_2, A\alpha_3, \dots, A\alpha_n$
  - Produksi tidak rekursif kiri  $\beta_1, \beta_2, \beta_3, \dots, \beta_m$
- Ganti setiap produksi dalam bentuk  $A_i \rightarrow A_j\alpha$  dengan produksi  $A_i \rightarrow \beta_1 \mid \beta_1 A' \mid \beta_2 \mid \beta_2 A' \mid \beta_3 \mid \beta_3 A' \mid \dots \mid \beta_m \mid \beta_m A'$ , Dan  $A' \rightarrow \alpha_1 \mid \alpha_1 A' \mid \alpha_2 \mid \alpha_2 A' \mid \alpha_3 \mid \alpha_3 A' \mid \dots \mid \alpha_n \mid \alpha_n A'$
- Lakukan langkah 1 dan 2 untuk semua aturan produksi pada CFG

- Lakukan testing untuk memastikan program berjalan dengan benar
- Dokumentasikan form/interface, prosedur, fungsi, input dan luaran program yang dibuat dalam kertas A4, diketik dengan huruf times new roman 12 dengan 1 spasi.

## C. Pretest Praktikum

Buatlah bentuk CNF dari CFG sebagai berikut menggunakan program yang telah anda buat.

$$S \rightarrow SABC \mid Ba \mid ab \mid a \mid c$$

#### D. Postest Praktikum

Gunakan program yang anda buat untuk membuat CNF pada kasus CFG berikut:

**S** → **Sba** | **Abc** | **b**

**A** → **AbB** | **ab** | **Ac** | **d**

**B** → **BcA** | **b**

#### E. Referensi

Maza, M. M. (2004). Elimination of left recursion. Retrieved from <http://www.csd.uwo.ca/~moreno//CS447/Lectures/Syntax.html/node8.html>

Moore, R. C. (2001). Removing Left Recursion from Context-Free Grammars. Revised version of paper appearing in Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics, ANLP-NAACL 2000.

Utdirartamo, F. (2005). *Teori Bahasa dan Otomata*. Yogyakarta: Penerbit Graha Ilmu.

Weimar, W., Leach, K., Thomason, Wi., Recto, R., Roberts, J., & Leath, H. (2015). Eliminating Immediate Left Recursion. Retrieved from <http://www.cs.virginia.edu/~cs415/reading/RemovingLeftRecursion.pdf>

<b>Nilai</b>	<b>Yogyakarta, .....</b>  <b>Paraf asisten</b>  <.....>
--------------	---

**Jawaban Posttest**



## ***Greibach Normal Form (GNF)***

Pertemuan	:	X
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan <i>interface</i> Greibach Normal Form (GNF) 2. Mahasiswa mampu memahami algoritama dari proses <i>GNF</i>
Indikator	:	1. Mahasiswa mampu membuat <i>interface GNF</i> dengan menggunakan <i>visual programming</i> atau <i>bahasa pemrograman lain yang dikuasai</i> 2. Mahasiswa mampu membuat prosedur/fungsi untuk GNF

### **A. Dasar Teori**

Greibach Normal Form (GNF) merupakan bentuk normal yang memiliki banyak konsekuensi teoritis dan praktis (Utdirartamo, 2005). Sebuah CFG dikatakan memenuhi bentuk normal greibach (GNF) apabila setiap produksi memenuhi bentuk (Gautam, 2011).

$$A \rightarrow a\alpha, \text{ dimana } a \in T \text{ dan } \alpha \in V^*$$

Sehingga dapat disimpulkan bahwa CFG yang memeneuhi GNF pada ruas kanan ( $\beta$ ) diawali oleh maksimum satu symbol terminal dan selanjutnya berisi symbol-simbol variable, atau hanya terdiri dari sebuah symbol terminal saja. Agar CFG dapat diubah ke GNF, maka harus memenuhi syarat, yaitu (Utdirartamo, 2005) :

1. Sudah dalam bentuk normal Chomsky
2. Tidak bersifat rekursif kiri
3. Tidak menghasilkan  $\epsilon$

Contoh aturan produksi CFG yang sudah memenuhi GNF:

$$\begin{aligned} S &\rightarrow aBC \mid b \\ B &\rightarrow aC \mid aB \mid b \\ C &\rightarrow cBDB \mid b \end{aligned}$$

## B. Petunjuk Praktikum

1. Jalankan *visual programming* atau bahasa pemrograman lain yang dikuasai
2. Buat desain form input dan luaran untuk proses pembentukan GNF
3. Buat prosedur/fungsi untuk pembentukan GNF berdasarkan algoritma sebagai berikut:

1. Ubah nama semua symbol variable menjadi urutan  $A_1, A_2, A_3, \dots, A_n$ , dimana  $A_1 = S$  (symbol awal)
2. Seluruh aturan produksi yang ruas kanannya diawali dengan symbol Variabel, dapat dituliskan dalam bentuk  $A_h \rightarrow A_i \gamma$  dimana  $h < i$  (Rekursif kiri sudah dihilangkan),  $\gamma$  berupa symbol variable.
3. Jika  $h < i$ , aturan produksi ini sudah benar (tidak perlu diubah)
4. Jika  $h > i$ , aturan produksi belum benar. Lakukan substitusi berulang-ulang terhadap  $A_i$  (ganti  $A_i$  pada produksi ini dengan ruas kanan produksi dari variable  $A_i$ ) sampai diperoleh aturan produksi dalam bentuk  $A_h \rightarrow A_p \gamma$  (dimana  $h \leq p$ ), dimana :
  - 1) Jika  $h = p$ , lakukan penghilangan rekursif kiri
  - 2) Jika  $h < p$ , aturan produksi sudah benar
5. Jika terjadi penghilangan rekursif kiri pada tahap 4, sejumlah symbol variable yang muncul dari operasi ini dapat disisipkan pada urutan variable semula di mana saja asalkan tidak ditempat sebelum  $A_h$ .
6. Setelah langkah 4 dan 5 dilakkan, maka aturan-aturan produksi yang ruas kanannya dimulai

simbol variable sudah berada dalam bentuk urutan yang benar.

$$A_x \rightarrow A_y\gamma \text{ (dimana } x < y\text{)}$$

Produksi-produksi yang lain ada dalam bentuk:

$$A_x \rightarrow a\gamma \text{ (a=symbol terminal)}$$

$$B_x \rightarrow \gamma$$

( $B_x$  = symbol variable yang muncul akibat operasi penghilangan rekursif kiri)

7. Lakukan substitusi mundur mulai dari variable  $A_n$ ,  $A_{n-1}$ ,  $A_{n-2}$ ,... Dengan cara ini aturan produksi dalam bentuk  $A_x \rightarrow A_y\gamma$  dan  $B_x \rightarrow \gamma$  dapat diubah, sehingga ruas kanannya dimulai dengan symbol Terminal

4. Lakukan testing untuk memastikan program berjalan dengan benar
5. Dokumentasikan form/interface, prosedur, fungsi, input dan luaran program yang dibuat dalam kertas A4, diketik dengan huruf times new roman 12 dengan 1 spasi.

### C. Pretest Praktikum

Buatlah bentuk GNF dari CFG sebagai berikut (Utdirartamo, 2005) menggunakan program yang telah anda buat.

$$S \rightarrow CA$$

$$A \rightarrow a \mid d$$

$$B \rightarrow b$$

$$C \rightarrow DD$$

$$A \rightarrow AB$$

### D. Postest Praktikum

Gunakan program yang anda buat untuk membuat CNF pada kasus CFG berikut:

**S → BC**

**A → DB | b**

**B → CA | a**

**C → AC | c**

**D → d**

### **E. Referensi**

Gautam, K. Das. (2011). Properties of Context-free Languages. Retrieved from <http://www.iitg.ernet.in/gkd/ma513/oct/oct18/note.pdf>

Utdirartamo, F. (2005). *Teori Bahasa dan Otomata*. Yogyakarta: Penerbit Graha Ilmu.

<b>Nilai</b>	<b>Yogyakarta, .....</b>  <b>Paraf asisten</b>  <.....>
--------------	---

**Jawaban Posttest**