

LAPORAN TUGAS BESAR
Tugas Besar IF 2220 Teori Bahasa Formal dan Otomata
Permainan “Tic-Tac-Toe” dengan Finite Automata



Oleh

Irfan Sofyana Putra – 13517078

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2018

BAB I

Deskripsi dan Batasan Masalah

A. Deskripsi Masalah

Tic-tac-toe adalah sebuah permainan untuk dua orang yang secara bergiliran saling membuat huruf X dan O di dalam sebuah kotak berukuran 3x3. Pemain pertama yang berhasil meletakkan 3 buah tanda 'X' atau 'O' secara vertikal, horizontal, atau diagonal akan dinyatakan menang.

Gambar berikut ini mendeskripsikan permainan tic-tac-toe yang dimenangkan oleh pemain yang menggunakan tanda 'X'



Pada tugas besar mata kuliah IF 2220 Teori Bahasa Formal dan Otomata ini, kita diminta untuk merancang sebuah permainan tic-tac-toe dengan ketentuan permainan ini dimainkan oleh seorang *user* dan sebuah komputer. Adapun hasil permainan tic-tac-toe ini adalah komputer **tidak diperkenankan** untuk kalah.

Permainan tic-tac-toe yang dibuat harus menggunakan finite automata. Sehingga pada saat aplikasi dibuka pertama kali, maka aplikasi akan membaca sebuah file berisi daftar *states*, *state* awal, *state* akhir, daftar symbol, dan *transition function*. Adapun logika *state-machine* tidak boleh di-*hardcode* ke program secara langsung.

B. Batasan Masalah

Pada saat permainan dimulai, *user* diminta memilih untuk memulai permainan terlebih dahulu atau komputer terlebih dahulu. *User* akan menggunakan tanda 'X' sedangkan komputer akan menggunakan tanda 'O'. Setiap pemain yang mulai terlebih dahulu, akan selalu menempatkan simbolnya pada tengah papan (posisi nomor 5). Hasil yang diinginkan pada permainan tic-tac-toe ini adalah komputer **tidak diperkenankan** untuk kalah. Sebagai kesepakatan bersama, posisi dari papan permainan tic-tac-toe ini dapat diberi nomor seperti pada gambar berikut ini:

1	2	3
4	5	6
7	8	9

BAB II

States dan Penjelasan DFA

A. Konsep & *States* Permainan

Untuk menyelesaikan permainan tic-tac-toe dengan hasil komputer tidak boleh kalah, maka *state* yang saya gunakan adalah berupa **langkah-langkah optimal** yang akan membawa permainan menuju kemenangan bagi komputer atau berakhir dengan seri. *State* tersebut saya dapatkan dengan manual yaitu mencoba semua kemungkinan masukan dari *user*, lalu pilih langsung satu tempat yang optimal (tidak terdapat kemungkinan kalah). Jika terdapat beberapa tempat yang optimal, maka boleh pilih yang mana saja.

Adapun *finite automata* yang saya gunakan adalah dua buah DFA. Dua buah DFA berdasarkan kasus ketika *user* bermain pertama dan ketika komputer bermain pertama. *States* yang saya gunakan pada DFA ini adalah berupa string dengan panjang 9 atau panjang 10. String dengan panjang 9 melambangkan sebuah *states* yang bukan *final states*, sedangkan *states* dengan panjang 10 adalah sebuah *final states*.

Untuk *states* selain *final states*, maka bentuk umum *states* tersebut adalah $C_1C_2C_3C_4C_5C_6C_7C_8C_9$ dimana untuk setiap $i, 1 \leq i \leq 9$ berlaku C_i adalah salah satu di antara karakter 'A', 'O' atau 'X'. Jika $C_i = 'A'$, maka papan tic-tac-toe dengan posisi nomor- i masih kosong (belum ada karakter 'O' dan 'X') sementara jika $C_i = 'O'$ atau $C_i = 'X'$, maka papan tersebut sudah diisi oleh komputer/*user*.

Untuk *final states*, bentuk umum dari *states* tersebut adalah $C_1C_2C_3C_4C_5C_6C_7C_8C_9C_{10}$. Dimana untuk setiap $i, 1 \leq i \leq 9$, C_i melambangkan keadaan papan dari tic-tac-toe yang sedang digunakan (sama seperti *states* selain *final states*) sedangkan untuk C_{10} adalah salah satu di antara '*' atau '#'. Jika $C_{10} = '*'$, maka *states* tersebut adalah *states* yang berakhir dengan kemenangan bagi komputer sedangkan jika $C_{10} = \#$, maka *states* tersebut melambangkan kondisi permainan tic-tac-toe dengan hasil seri/imbang.

Contoh kondisi permainan & *states*:

X		O
	O	
	X	O

Kondisi permainan tic-tac-toe pada gambar di samping bisa dideskripsikan dengan sebuah *state* yaitu XAOAOAAXO.

	X	X
O	O	O

Kondisi permainan tic-tac-toe pada gambar di samping bisa di deskripsikan dengan sebuah *state* yaitu AXXOOOAAA*. Perhatikan bahwa kondisi permainan tersebut merupakan kondisi menang bagi komputer, sehingga *state* yang digunakan akan mendeskripsikan *final state* (diberi '*')

Untuk jumlah *states* yang saya gunakan pada DFA saat *user* bermain pertama adalah 62 *states* dengan 32 *states* diantaranya adalah *final states*. Adapun saat komputer bermain pertama, jumlah *states* yang saya gunakan adalah 104 *states* dengan 77 *states* diantaranya adalah *final states*.

Saya menggunakan bahasa pemrograman C untuk mengimplementasikan permainan Tic-tac-toe ini. Adapun *states* yang saya gunakan pada 2 DFA tersebut saya simpan dalam struktur data yang sama.

B. DFA Saat User Bermain Pertama

Saat *user* bermain pertama, komponen DFA yang saya gunakan adalah sebagai berikut:

1. Alphabet yang dipakai {1, 2, 3, 4, 5, 6, 7, 8, 9}
2. *User* akan mengisi papan dengan posisi nomor 5 terlebih dahulu. Setelah itu, komputer akan langsung mengisi papan dengan posisi nomor 1. Oleh karena itu *start state* saat *user* bermain pertama adalah OAAAXAAAA.
3. *Final states* saat *user* bermain pertama ada 32 *states*. *State* ini akan menghasilkan permainan dengan kemenangan bagi komputer atau didapatkan hasil seri.
4. *States* total yang digunakan pada saat *user* bermain pertama ada 62 *states*
5. *Transition function*, berisi kondisi *state* setelah menerima *input* dari *user* atau balasan dari komputer dengan mengambil langkah yang optimal.

```
OAAAXAAAA OXAAXA0A OXXAXA0A OXXAX00X OXXX000X# OXXAXX000* OXXXA000* OXAXX0A0A OXXX000A OX0X0X0A
OX0X0X0X# OX0X0A0X OXA0XXA0A OX00XX0A OX00XX0X# OX0X0X0A* OXA0XX00X* OX0AXAX0A OX00XAX0X OX0AXA0X
OX00XXA0X OAXAXA0A OAXXX00A OXX0X0XA OXX0X00X# OXX0X0AX OXX0XA0AA* OAX0X0AA* OAX0XAX0A* OAX0XAX0X*
OAAAX0AAA OAOXX0XA OX0X0XAX* OAOXX0X0* OAOXX0X0X OAOXX0AXA OXX0X0XA* OXX0X0AXX* OAA0X0A0X OAXXX000X
OAA0X0AAA OXA0XX0AA* OAOXX0XA OXX0XX0A* OXX0XX0X OXX0XX0X* OAA0XX0AX* OAOAXAXAA OXX0XAXAA* OXX0XXAA*
OXX0XAXX* OXX0XAXX* OAAAXAAX OXXAXA0X OXX0X0XA* OXX0XAXX* OXX0XAXA* OXX0XAXA* OXX0XAXX* OAOAXAAX
OXX0XAXX* OXX0XAXX*
```

States yang digunakan pada DFA saat *user* bermain pertama.

```
OXXX000X# OXXAXX000* OXXXA000* OX0X0X0X# OX00XX0X# OX0X0X0A* OXA0XX00X* OXX0X00X# OX0XA0AA* OAX0X0AA*
OAX0XAX0A* OAX0XAX0X* OX0X0XAX* OAOXX0X0* OXX0X0XA* OXX0XAXX* OXA0XX0AA* OXX0XX0A* OXX0XXAX* OAOXX0XA*
OAOXX0AX* OXX0XAXA* OXX0XXAA* OXX0XXAA* OXX0XXAX* OXX0X0XA* OXX0XAXX* OXX0XAXA* OXX0XAXA* OXX0XAXX*
OXX0XAXX* OXX0XAXX*
```

Final states pada DFA saat *user* bermain pertama

[illegible]

Transition table pada DFA saat *user* bermain pertama

Contoh cara membaca *transition table*:

Misal *states* sekarang adalah **OAAAXAAAA**(*start state*), lalu *user* input 3, maka *states* tersebut akan berpindah menuju *states* **OAXAXAOAA**

Contoh masukan pada DFA *user* bermain pertama:

1. *User harus menginputkan posisi 5* terlebih dahulu. Jika pada awalnya *user* tidak menginputkan 5, maka input akan diulang hingga *user* menginput 5.

	X	

2. Setelah *user* menginput 5, maka komputer akan langsung memilih posisi 1 untuk diberi tanda 'O', sehingga **states awal** pada saat *user* bermain pertama adalah **OAAAXAAAA**.

O		
	X	

3. Kemudian *user menginput 2*, maka *states* akan berpindah menuju *states OXAAXAAOA*.

O	X	
	X	
	O	

4. Jika posisi yang diinputkan oleh *user* telah diisi oleh suatu simbol, maka *states* akan **kembali kedirinya sendiri**. Misal *user* kembali menginputkan 2, maka *states* permainan tetap pada *states OXAAXAAOA*. Setelah itu *user* akan kembali diminta menginput posisi untuk simbol 'X' ditempatkan. Kemudian kali ini *user menginput 3*. Maka *states* akan berpindah menuju *states OXXAXAOOA*

O	X	X
	X	
O	O	

5. Kali ini *user menginput 4*, maka *states* akan berpindah menjadi **OXXXXAOOO***. Pada kasus ini permainan berakhir karena komputer berhasil memenangkan permainan dengan menempatkan simbol 'O' pada posisi 9.

O	X	X
X	X	
O	O	O

6. Jika DFA sudah mencapai final *states*, maka *transition* dari *states* tersebut akan kembali menuju dirinya sendiri. Artinya *states* tersebut sudah diterima dan tidak akan berganti. Di dalam permainannya sendiri, maka permainan akan selesai jika sudah mencapai *final states*.

C. DFA saat Komputer Bermain Pertama

Saat komputer bermain pertama, komponen DFA yang saya gunakan adalah sebagai berikut:

1. Alphabet yang dipakai {1, 2, 3, 4, 5, 6, 7, 8, 9}
2. Komputer akan mengisi papan dengan posisi nomor 5 terlebih dahulu. Sehingga *start state* saat komputer bermain pertama adalah **AAAAOAAAA**
3. *Final states* saat komputer bermain pertama ada 77 *states*. *State* ini akan menghasilkan permainan dengan kemenangan bagi komputer atau didapatkan hasil seri.
4. *States* total yang digunakan pada saat komputer bermain pertama ada 104 *states*
5. *Transition function*, berisi kondisi *state* setelah menerima *input* dari *user* atau balasan dari komputer dengan mengambil langkah yang optimal.

```
AAAAOAAAA XAAOAAAA XAOAOAAO XAAOAAO XAOOXXAO XAAOXXAO XAAOAAO XAOOAAO XOXOAAO XOXOAAO
AXAOAAAA OXAOAXAA AAXOAAAA AAXOAXAA XOXOAXAA AAOXAAAA OAOXAXAA AAOAXAAA OAAOAXAA AAAOAXAA
OAXOAXAA OAXOAXOX AAAOAXAA AAOXOAXA AAAOAXOAX AAXOAXOAX AOXOAXOAX XAOXOAXA* XAOXOAXA* XXAOXOAXA*
AXXOAXOAX* AAXOAXOAX* AXAOXOAX* AXAOXOAX* AXAOXOAX* OXXOAXOAX* OXAOXOAX* OXAOXOAX* AAXOAXOAX*
AAXOAXOAX* XAXOAXOAX* OAXOAXOAX* OAXOAXOAX* XOXOAXOAX# XOXOAXOAX* AXOXOAXOAX* AAOXOAXOAX* AAOXOAXOAX*
OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* XAOXOAXOAX* AOXOAXOAX* AOXOAXOAX* AOXOAXOAX* AOXOAXOAX*
OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX*
AAOAXOAX* AAOAXOAX* XAOXOAXOAX* XAOXOAXOAX* XAOXOAXOAX* AAOXOAXOAX* AAOXOAXOAX* AAOXOAXOAX* AAOXOAXOAX*
XAXOAXOAX* AAXOAXOAX* AAXOAXOAX* XOXOAXOAX* OXXOAXOAX# XOXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX*
XXAOAXOAX* XAXOAXOAX* XAAOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX*
XXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX*
```

States yang digunakan pada DFA saat komputer bermain pertama

```
XAOXOAXOAX* XAOXOAXOAX* XAOXOAXOAX* AAXOAXOAX* AAXOAXOAX* AAXOAXOAX* AAXOAXOAX* AAXOAXOAX* AAXOAXOAX*
OXAOXOAXOAX* OXAOXOAXOAX* AAXOAXOAX* AAXOAXOAX* XAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX*
AAOXOAXOAX* AAOXOAXOAX* AAOXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX*
AOAXOAXOAX* AOAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX* OAXOAXOAX*
OXOXOAXOAX# OXXOAXOAX* AAOXOAXOAX* AAOXOAXOAX* AAOXOAXOAX* AAOXOAXOAX* AAOXOAXOAX* AAOXOAXOAX* AAOXOAXOAX*
AAOAXOAXOAX* AAOAXOAXOAX* AOXOAXOAX* XAXOAXOAX* AAXOAXOAX* AAXOAXOAX* XOXOAXOAX* OXXOAXOAX# XOXOAXOAX*
XAOXOAXOAX* XAOXOAXOAX* XAOXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX*
XAXOAXOAXOAX* XAXOAXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX* XOXOAXOAX*
```

Final states pada DFA saat komputer bermain pertama

- | | | |
|--|---|--|
| | | |
| | O | |
| | | |

2. Selanjutnya adalah giliran *user*. Misalkan *user* **menginput 2** maka *states* akan berpindah menuju *states* **AXAOOAAAA**.

	X	
O	O	

3. Jika *user* menginputkan posisi yang sudah terisi oleh suatu simbol, maka *states* tersebut tidak akan berubah. Misal ketika *user* menginput 2, maka *states* tetap pada *states* **AXAOOAAAA**. Selanjutnya *user* kembali diminta untuk menginput posisi yang ingin diisi. Misalkan pada saat sekarang *user* **menginput 6**, maka *states* akan berpindah menuju *states* **OXAOOXAAA**.

O	X	
O	O	X

4. Pada *states* **OXAOOXAAA**, semua *input* dari *user* akan menghasilkan kemenangan bagi komputer. Misalkan *user* **menginput 3**, maka *states* akan berpindah menuju *states* **OXXOOXAAO***.

O	X	X
O	O	X
		O

5. *States* **OXXOOXAAO*** adalah sebuah *states* yang menandakan komputer akan memenangkan permainan karena komputer telah mendapatkan 3 simbol 'O' yang ditulis secara diagonal
6. Jika DFA sudah mencapai final *states*, maka *transition* dari *states* tersebut akan kembali menuju dirinya sendiri. Artinya *states* tersebut sudah diterima dan tidak akan berganti. Di dalam permainannya sendiri, maka permainan akan selesai jika sudah mencapai *final states*.

Source Code

Bahasa pemrograman yang saya gunakan untuk mengimplementasikan permainan Tic-Tac-Toe ini adalah bahasa C. Selain itu, saya juga menggunakan sebuah file eksternal yang menyimpan informasi mengenai DFA yang digunakan pada program.

A. File eksternal

File eksternal yang saya gunakan memiliki nama “FileStates.txt”. File ini berisi informasi mengenai daftar *states*, *initial / start states*, *final states*, daftar simbol, dan *transition table* dari DFA yang digunakan oleh program. Dua DFA yang saya buat untuk menyelesaikan program Tic-Tac-Toe ini **disimpan** dalam satu file ini. (tidak dibedakan).

Screenshot file eksternal (FileStates.txt) :

```

DaftarStates:
AAAAOAAAA XAAA0AAAA XAA0A0AA0 XAA00AXA0 X0A00XXA0 XAAA0X0A0 XAAAX0A0A0 X0X0A0AA0 X0X00AA0X0
XAA00AAAA 0XAA0XXAA XAA00AAAA AAX00XAA0 X0X00XAA0 AA0X0AAAA A0A0X0XAA 00AA0XAXA AAAA00XAA
0AAAX00XAA 0AAAX00X0X AAAA00AXA AA0X00AXA AAAA0A0AX AAXA000AX A0XXX000AX XA0X0A0AA* XA0A0A0AX* XXA000AAA*
AXX000AAAA* AAX000XAA* AXA000XAA* AXA000AXA* AXA000AAX* 0XX00XAA0* 0XA00XXA0* 0XA00XAX0* 0XA00X0AX* AAX000AAX*
AAX000AXA* XAX000AAA* 0AX000XAA* 0AX00XAX0* X0X00X0X0# X0X00XX00* AX0X0A0AA* AA0X0X0AA* AA0X0A0XA* AA0X0A0AX*
00X0X0AXX* 000X0AXXA* 000X0XXAA* 0X0X0AXA0* X0A00XAA0* A0AX0AXA0* A0A0X0X0A* A0A0A0X0A* A0A0A0AXX*
000A0X0XA* 000X0XAXA* 00XA0XAX0* XAA000XAA* 0XAX00XAA0* 0AXX00XAA0* 0AAX00XX0* 0X0X00X0X# 00XX00X0X* AAA000AXX*
AAA000XXA* AAA000XAX* XAA000AXA* XA0X000XA* AX0X000XA* AA0X00XX0* AA0X000XX* AA0A0A0AX* AA0A0X0AX* AX0A0A0AX*
XAA0000AX* AXX0000AX* AAX0000X* X0XX0000X* 00XX000XX# X0000XX0# XX0A00AX0* XA0X00AX0* XA0A0X0X0* XA0A00XX0*
XXA000XAA0* XAX000XAA0* XAA000X0X* XX0A0X0AA0* XAXA0X000* XA0X0X0AA0* XA0X0A0X0* XAXX0A000* XXAX0A000* XX0X00AA0*
XX0A00XAA0* X0X0A0AX00* X0X0AXA00* X0XX0AA00* 0AAAXAAAA 0AAXA0A0A 0XXAXA00A 0XXAX000X 0XX0X000X# 0XXAXX000*
00XX0A000* 0XAXX0A0A 0XX0X000A 0X0XX0X0A 0X0XX0X0X# 0XX0X0A0X 0XAX0X0A 0X00XX00A 0X00XX0X0# 0XX0X00A
0XA0XX00X* 0X0AXAX0A 0X00XAX0X 0X0AXAA0X 0X00XXA0X 0AXAXA0AA 0AXXX00AA 00XX00XA 00XX00XX# 00XX00AX
0XX0XA0AA* 0AX0XX0AA* 0AX0XA0AX* 0AX0XA0AX* 0AAXX0AAA 0A0XX0XAA 0X0XX0XAA0* 0A0XX0XX0* 0A0XX0X0X 0A0XX0AXA
000XX0XXA* 000XX0AXX* 0AAXX0A0X 0AAXX000X 0AAX0XAAA 0XA0XX0AA* 0A00XXAA* 0000XX0XA* 0000XXAX* 0AA0XX0XA*
0A0XX0AX* 0A0AXAXAA 000XXAXA* 00A0XXAA* 00A0AXXA* 000AXAXA* 00AAXAA 00XAXA0X 0X0XX0XA* 00X0XA0XX*
000XXAAXA* 00A0XXAXA* 00A0AXAAX* 0A0AXAAAX 00A0XXAAX* 000XXAAX*

InitialStates:
AAAAOAAAA 0AAAXAAAA

FinalStates:
XAX0X0A0AA* XAX0A0A0AX* XXA000AAA* AXX000AAA* AAX000XAA* AXA000XAA* AXA000AXA* AXA000AAX* 0XX00XAA0* 0XA00XXA0*
0XA00XAX0* 0XA00X0AX* AAX000AAX* AAX000AXA* XAX000AAA* 0AX00XXAA0* 0AX00XAX0* X0X00X0X0# X0X00XX00* AX0X0A0AA*
AAX0X0XAA* AA0X0A0AX* AA0X0A0AX* 000X0AXAX* 000X0AXXA* 000X0XXAA* 0X0X0AXA0* 0XA00XAA0* A0XAX0A0A* A0AX0XAA0*
A0A0X0X0A* A0A0X0AXX* 000A0XAXX* 000A0XX0A* 000X0AXA* 00XAXA0A* XAA000XAA* 0XAX00XAA0* 0AXX00X0A* 0AXX00XX0*
0X0X00X0X# 00XX00X0X* AAA000AXX* AAA000XXA* AAA000XAX* XAA000AXA* XA0X000XA* AX0X000XA* AA0X00XX0* AA0X000XX*
AA0A0A0AX* AA0A0X0AX* AX0A0A0AX* XAX0000AX* AAX0000AX* AAX0000XX* X0XX0000X* 00XX000XX# X0000XX0# XX0A00X0X*
XAX0X0A0X* XAX0X0X0X* XA0A00X0X* XXA000XAA* XAX000XAA* XAA000XX0* XX0A0X0AA* XAXAX0000* XA0X0X0A0* XA0X0A0X0*
XAXX0A000* XXAX0A000* XX0X00AA0* XX0A00XAA0* X0XAXA000* X0XAXA000* X0XX0A000* 0XX0X000X# 0XXAXX000* 0XX0X000*
0X0XX0X0X# 0X00XX0X0# 0X0X0X00A* 0XA0X000X* 00XX00XX# 0X0X0A0A* 0AX0X0A0A* 0AX0XA0XA* 0AX0X0AAX* 0X0XX0XAA0*
0A0XX0XX0* 000X0X0XA* 000X0XAXX* 0XA0XX0AA* 0000XXAA* 0000XXAX* 0AA0XX0XA* 000XXAXAA* 000AXXXAA*
00AAXAXXA* 000AXAXAA* 00X0XX0XA* 000X0X0AX* 000X0AXX* 000AXAXXA* 00AAXAXA* 00AAXAAXX* 00AAXAAXX* 000XXAAXX*

```

Daftar *states*, *Initial/start states*, *Final states*

[illegible]


```

FILE *filestates, *ftmp;
int counter, NStates, NFinal, NInitial, NSimbol, Nlewat;
bool isPlaced;

/*Prosedur untuk membaca file yang berisi states*/
/*I.S : File "FileStates.txt" belum dibaca */
/*F.S : File "FileStates.txt" sudah dibaca dan datanya disalin pada variabel yang
beresuaian */

void BacaFileStates(){
    /*Membuka file yang berisi daftar states, final states, initial states,
    Daftar simbol, dan transition Table*/

    filestates = fopen("FileStates.txt", "r");

    fscanf(filestates, "%s", &arr);
    if (strcmp(arr, "DaftarStates:") == 0){
        /* Membaca semua daftar states yang digunakan pada program */
        NStates = 1;
        do{
            fscanf(filestates, "%s", &arr);
            if (strcmp(arr, "InitialStates:") == 0) break;
            else {
                strcpy(DaftarStates[NStates].States, arr);
                NStates++;
            }
        }while (strcmp(arr, "InitialStates:") != 0);
        NStates--;
    }
    if (strcmp(arr, "InitialStates:") == 0){
        /* Membaca semua initial states yang digunakan pada program */
        NInitial = 1;
        do{
            fscanf(filestates, "%s", &arr);
            if (strcmp(arr, "FinalStates:") == 0) break;
            else {
                strcpy(InitialStates[NInitial].States, arr);
                NInitial++;
            }
        }while (strcmp(arr, "FinalStates:") != 0);
        NInitial--;
    }
    if (strcmp(arr, "FinalStates:") == 0){
        /* Membaca semua final states yang digunakan pada program */
        NFinal = 1;
        do{
            fscanf(filestates, "%s", &arr);
            if (strcmp(arr, "DaftarSimbol:") == 0) break;
            else {
                strcpy(FinalStates[NFinal].States, arr);
                NFinal++;
            }
        }while (strcmp(arr, "DaftarSimbol:") != 0);
        NFinal--;
    }
}

```

```

    if (strcmp(arr, "DaftarSimbol:") == 0) {
        /* Membaca semua daftar simbol yang digunakan pada program */
        NSimbol = 1;
        do{
            fscanf(filestates, "%s", &arr);
            if (strcmp(arr, "TransitionsTable:") == 0) break;
            else {
                strcpy(DaftarSimbol[NSimbol].States, arr);
                NSimbol++;
            }
        }while(strcmp(arr, "TransitionsTable:") != 0);
        NSimbol--;
    }
    if (strcmp(arr, "TransitionsTable:") == 0){
        /*Membaca transition table dari states-states yang digunakan pada
        program */
        NStates = 1;
        counter = 0;
        while (fscanf(filestates, "%s", &arr) != EOF){
            if (counter > 9) {
                counter = 0;
                NStates++;
            }
            strcpy(TransitionStates[NStates][counter].States, arr);
            counter++;
        }
    }
}

/*Prosedur untuk mencetak papan permainan yang sedang berlangsung */
/*I.S : kondisi permainan dideskripsi kan dengan array of char */
/*F.S : Tercetak kondisi papan permainan pada layar */
void cetakPapan(char arr[]){
    printf("Kondisi Papan Tic-Tac-Toe Sekarang adalah: \n");
    printf("-----\n");
    for (int i = 0; i < 9; i++){
        if (i%3 == 0){
            if (arr[i] == 'A')
                printf("| |");
            else if (arr[i] == 'O')
                printf("|O|");
            else
                printf("|X|");
        }else if (i%3 == 1){
            if (arr[i] == 'A')
                printf(" |");
            else if (arr[i] == 'O')
                printf("O|");
            else
                printf("X|");
        }else{
            if (arr[i] == 'A')
                printf(" |\n");
            else if (arr[i] == 'O')
                printf("O|\n");
            else
                printf("X|\n");
        }
    }
    printf("-----\n");
}

```



```

/*Mencari states pada DaftarStates */
/*Pre-kondisi : s[] adalah states yang terdefinisi */
/*Hasil      : indeks dimana states s[] berada */

int cariStates(char s[]){
    for (int i = 1; i <= NStates; i++){
        if (strcmp(DaftarStates[i].States, s) == 0) return i;
    }
}

/*Mengecek apakah posisi "pos" pada papan permainan sekarang masih kosong atau
tidak*/
/*Pre-kondisi : s[] adalah sebuah states yang terdefinisi dan pos adalah posisi
yang valid pada papan */
/*Hasil : True, jika s[pos] masih kosong dan false, jika sudah terisi */

bool CanPlaced(char s[], int pos){
    if (s[pos] != 'A') return false;
    return true;
}

/*Prosedur untuk memainkan permainan tic-tac-toe dimana komputer akan bermain
pertama */
/*I.S : komputer akan bermain pertama */
/*F.S : komputer tidak akan kalah dalam permainan*/

void comFirst(){
    system("cls");
    printf("Komputer bermain terlebih dahulu!\n");
    printf("\n");

    //now adalah states dari permainan yang sedang berlangsung
    strcpy(now, TransitionStates[1][0].States);
    int x = 0;
    int Nlewat = 1;
    //selama states sekarang belum final states
    while (strlen(now) == 9){
        if (x == 0) {
            //Giliran komputer yang bermain
            printf("Giliran Komputer: \n");

            //cetak papan permainan
            cetakPapan(now);
            printf("\n");

            //salin states sekarang ke array yang berisi states yang //sudah
            dilewati selama permainan
            strcpy(StatesLewat[Nlewat].States, now);
            Nlewat++;

            //Giliran selanjutnya adalah user
            x = 1-x;
        }
        else {
            //Giliran user yang bermain
            printf("Giliran Anda: \n");

            //meminta masukan dari user untuk memilih posisi dimana //simbol
            x akan ditempatkan
            int bil;
            printf("Masukkan tempat yang ingin diisi(1-9): ");
            scanf("%d", &bil);
            bil--;
        }
    }
}

```

```

        //jika posisi yang diinputkan belum diisi, maka isi //tempat itu
        dengan 'X'
        if (CanPlaced(now, bil)) {
            //mencari states sekarang di daftarStates
            int indeks = cariStates(now);

            //isi posisi yang diinputkan user dengan simbol 'X'
            bil++;
            now[bil-1] = 'X';

            //cetak papan permainan sekarang
            cetakPapan(now);
            printf("\n");

            //States sekarang berubah sesuai dengan transition //table
            dan input dari user
            strcpy(now, TransitionStates[indeks][bil].States);

            //Giliran selanjutnya adalah komputer
            x = 1-x;
        }
        //posisi yang diinputkan oleh user sudah ditempati, maka
        //states kembali ke dirinya sendiri.
        //input dari user akan diulang
        else {
            //salin states sekarang ke array yang berisi states //yang
            sudah dilewati selama permainan

            strcpy(StatesLewat[Nlewat].States, now);
            Nlewat++;

            printf("Tolong fokus ya:( udah diisi itu tempatnya:(\n");
            printf("ulangi lagi yaa\n\n");
        }
    }
}
/*Permainan berakhir */

//jika karakter terakhir dari final states adalah '*', maka komputer menang
if (now[9] == '*') printf("Mohon maaf, Anda Kalah!\n");

//jika karakter terakhir dari final states adalah '#', maka permainan
//berakhir seri

else printf("Permainan berakhir seri!\n");

//cetak kondisi papan permainan
cetakPapan(now);

//salin final states ke array yang berisi states yang sudah dilewati selama
//permainan

strcpy(StatesLewat[Nlewat].States, now);

//mencetak semua states yang telah dilewati selama permainan berlangsung ke
//layar
printf("\n");
printf("States yang dilewati: \n");
for (int i = 1; i <= Nlewat; i++)
    printf("%d. %s\n", i, StatesLewat[i].States);
}

```

```

/*Prosedur untuk memainkan permainan tic-tac-toe dimana user akan bermain pertama
*/
/*I.S : user akan bermain pertama */
/*F.S : komputer tidak akan kalah dalam permainan*/

void playerFirst(){
    system("cls");
    printf("Anda bermain terlebih dahulu!\n");
    printf("\n");
    int no;

    //User akan diminta untuk mengisi posisi nomor 5 terlebih dahulu sesuai
    //dengan batasan permasalahan

    printf("Isi posisi nomor 5 terlebih dahulu ya\n");
    do{
        printf("Masukkan posisi yang ingin diisi: ");
        scanf("%d", &no);
        if (no != 5) printf("Tolong isi nomor 5 terlebih dahulu yaa :)\n");
    }while(no != 5);

    //Now adalah states dari permainan yang sedang berlangsung
    strcpy(now, TransitionStates[105][0].States);
    int x = 0;
    int Nlewat = 1;

    //Selama states belum final states atau permainan masih belum selesai
    while (strlen(now) == 9){
        if (x == 0) {
            //Giliran komputer yang bermain
            printf("Giliran Komputer: \n");

            //cetak papan permainan
            cetakPapan(now);
            printf("\n");

            //salin states sekarang ke array yang berisi states yang sudah
            //dilewati selama permainan
            strcpy(StatesLewat[Nlewat].States, now);
            Nlewat++;

            //Giliran selanjutnya adalah user
            x = 1-x;
        }
        else {
            //Giliran user yang bermain
            printf("Giliran Anda: \n");

            //meminta masukan dari user untuk memilih posisi dimana simbol x
            //akan ditempatkan

            int bil;
            printf("Masukkan tempat yang ingin diisi(1-9): ");
            scanf("%d", &bil);
            bil--;

            //jika posisi yang diinputkan belum diisi, maka isi tempat itu
            //dengan 'X'
            if (CanPlaced(now, bil)) {
                //mencari states sekarang di daftarStates
                int indeks = cariStates(now);

```

```

        //isi posisi yang diinputkan user dengan simbol 'X'
        bil++;
        now[bil-1] = 'X';

        //cetak papan permainan sekarang
        cetakPapan(now);
        printf("\n");

        //States sekarang berubah sesuai dengan transition table
        //dan input dari user
        strcpy(now, TransitionStates[indeks][bil].States);

        //Giliran selanjutnya adalah komputer
        x = 1-x;
    }
    //posisi yang diinputkan oleh user sudah //ditempati, maka states
    //kembali ke dirinya sendiri.
    //input dari user akan diulang

    else {
        //salin states sekarang kepada array yang berisi states
        //yang sudah dilewati selama permainan
        strcpy(StatesLewat[Nlewat].States, now);
        Nlewat++;

        printf("Tolong fokus ya :( udah diisi itu tempatnya:\n");
        printf("ulangi lagi yaa\n\n");
    }
}
}
/*Permainan berakhir */

//jika karakter terakhir dari final states adalah '*', maka komputer menang
if (now[9] == '*') printf("Mohon maaf, Anda Kalah!\n");
//jika karakter terakhir dari final states adalah '#', maka permainan
//berakhir seri
else printf("Permainan berakhir seri!\n");

//cetak kondisi papan permainan
cetakPapan(now);

//salin final states ke array yang berisi states yang sudah dilewati selama
//permainan

strcpy(StatesLewat[Nlewat].States, now);
printf("\n");

//mencetak semua states yang telah dilewati selama permainan //berlangsung ke
//layar

printf("States yang dilewati: \n");
for (int i = 1; i <= Nlewat; i++)
    printf("%d. %s\n", i, StatesLewat[i].States);
}

//Deklarasi prosedur untuk menampilkan interface permainan
void TampilanAwal();

//Prosedur menu games tic-tac-toe
//I.S : user akan memasuki permainan tic-tac-toe dan diminta untuk memilih memulai
permainan terlebih dahulu atau tidak
//F.S : games tic tac toe berakhir dengan komputer tidak akan kalah. Setelah itu
//user bisa memilih kembali ke main menu, bermain kembali, atau keluar langsung
//dari program

```



```

void games(){
    system("cls");
    printf("Anda memasuki permainan Tic-Tac-Toe!\n\n");
    int no;

    //Meminta masukan user apakah ingin bermain terlebih dahulu atau tidak
    printf("Pilih 1 jika Anda ingin main terlebih dahulu\n");
    printf("Pilih 2 jika Anda ingin komputer main terlebih dahulu\n");

    //Masukan akan diulang selama input dari user tidak valid
    do{
        printf("Masukkan Pilihan: ");
        scanf("%d", &no);
        if (no != 1 && no != 2) printf("Pilihan tidak valid!\n");
    }while (no != 1 && no != 2);
    printf("\n");

    //jika user memilih 1, artinya dia akan bermain terlebih dahulu
    if (no == 1) playerFirst();
    //jika user memilih 2, artinya komputer akan bermain terlebih dahulu
    else if (no == 2) comFirst();

    /*Permainan selesai*/

    //Menampilkan pilihan kepada user untuk kembali bermain, atau kembali ke main
    //menu, atau keluar langsung dari program
    printf("\n");
    printf("Pilih 1 untuk kembali ke main menu\n");
    printf("Pilih 2 untuk kembali bermain\n");
    printf("Pilih 3 untuk Keluar\n");

    //input dari user akan diulang jika input tidak valid
    do{
        printf("Masukkan Pilihan: ");
        scanf("%d", &no);
        if (no != 1 && no != 2 && no != 3) printf("Pilihan tidak valid!\n");
    }while (no != 1 && no != 2 && no != 3);

    //jika user memilih 1, maka program akan menampilkan main menu
    if (no == 1) TampilanAwal();
    //jika user memilih 2, maka program akan kembali menunjukan permainan tic-
    //tac-toe
    else if (no == 2) games();
    //jika user memilih 3, maka program telah selesai digunakan
    else return;
}

//Deklarasi prosedur untuk menampilkan cara bermain dari aplikasi tic-tac-toe yang
dibuat ini
void caraBermain();

//Prosedur untuk menampilkan interface dari aplikasi tic-tac-toe (main menu)
//I.S : Program akan menampilkan interface dari aplikasi tic-tac-toe
//F.S : Program akan menjalankan instruksi berdasarkan pilihan yang dipilih oleh
//user

void TampilanAwal(){

    //Menampilkan interface

```

```

printf("1. Cara bermain);
printf("2. Main);
printf("3. Keluar);
//program akan menerima pilihan dari user dan terus mengulanginya selama tidak
//valid
do {
    printf("Masukan Pilihan : ");
    scanf("%d", &no);
    if (no != 1 && no != 2 && no != 3) printf("Pilihan tidak valid\n");
}while (no != 1 && no != 2 && no != 3);

//jika user memilih 1, maka program akan menampilkan cara bermain dari
//aplikasi ini
if (no == 1) {
    printf("\n");
    caraBermain();
}
//jika user memilih 2, maka program akan langsung dialihkan untuk bermain
//tic-tac-toe
else if (no == 2){
    printf("\n");
    games();
}
//jika user memilih 3, maka program berhenti dan program telah selesai
//digunakan
else return;
}

//Prosedur untuk menampilkan cara bermain/cara menggunakan aplikasi tic-tac-toe yang
dibuat ini
//I.S : -
//F.S : Pada layar, akan diberikan informasi mengenai cara untuk
//menggunakan/bermain tic-tac-toe pada program ini

void caraBermain(){
    system("cls");

    //Menampilkan cara bermain dari tic-tac-toe pada program ini
    printf("Cara bermain Tic-Tac-Toe ini sangat mudah sekali!\n");
    printf("\n\n");
    printf("1. Pertama Pilih terlebih dahulu, Anda ingin main duluan atau
Komputer terlebih dahulu\n");

    printf("2. Anda akan menggunakan simbol X, sedangkan komputer akan
menggunakan simbol O\n");

    printf("3. Disaat giliran Anda bermain, Anda diminta untuk menginputkan
bilangan (1-9) yaitu posisi untuk menyimpan simbol X\n");

    printf("4. Posisi papan Tic-Tac-Toe bisa digambarkan menjadi gambar di bawah
ini: \n");

    printf("    ----- \n");
    for (int i = 0; i < 9; i++){
        if (i%3 == 0) printf("    ");
        if (i%3 == 0){
            printf("|%d|", i+1);
        }else if (i%3 == 1){
            printf("%d|", i+1);
        }else{
            printf("%d|\n", i+1);
        }
    }
    printf("    ----- \n");
    printf("\n");

```

```

//Meminta masukan user untuk kembali ke main menu atau langsung bermain
printf("Pilih 1 untuk kembali\n");
printf("Pilih 2 untuk bermain\n");
int no;

//Selama masukan dari user tidak valid, maka input akan terus diulang
do{
    printf("Masukan Pilihan: ");
    scanf("%d", &no);
    if (no != 1 && no != 2)
        printf("Pilihan tidak valid\n");
}while (no != 1 && no != 2);

//jika user memilih 1 maka program akan kembali ke main menu
if (no == 1) TampilanAwal();
//jika user memilih 2 maka permainan tic-tac-toe akan dimulai
else games();
}

//program utama dari games tic-tac-toe yang dibuat
int main(){
    //membaca informasi mengenai states permainan tic-tac-toe
    BacaFileStates();

    //Menampilkan main menu kepada user
    TampilanAwal();

    return 0;
}

```

Catatan pada *source code*:

1. Terdapat sedikit perbedaan penulisan *source code* pada laporan dengan *source code* sebenarnya. Bagian itu ada pada prosedur TampilanAwal(). Pada *source code* di laporan ini hanya diberi komen yaitu akan menampilkan sebuah *interface* program padahal sebenarnya *source code* akan menampilkan sekumpulan *ASCII code* untuk menampilkan tulisan “Tic-Tac-Toe games by irfansofyana”
2. Sekumpulan *states* yang dilewati adalah *start/initial state*, *states* ketika komputer menjawab respon dari gerakan *user*, *states* yang mengalami pengulangan (karena *input user* tidak valid), dan *final states*.