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 meletakan 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 papah (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 panjansg 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 $i, 1 \le i \le 9$ 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 $i, 1 \le i \le 9$, 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:

Х		0
	0	
	Х	0

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

	Χ	Χ
0	0	0

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-tactoe 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 **OAAAXAAA**.
- 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 OXAAXAAOA OXXAXAOOA OXXAXXOOOX OXXXXOOOX# OXXAXXOOO* OXXXXAOOO* OXAXXOAOA OXXXXXOOOA OXOXXOXOA
OXOXXOXOX# OXOXXOAOX OXAOXXAOA OXOOXXXOA OXOOXXXOX# OXXOXXOOA* OXAOXXOOX* OXOAXAXOA OXOOXAXOX OXOAXAAOX
OXOOXXAOX OAXAXAOAA OAXXXOOAA OOXXXOOXA OOXXXOOX# OXXXXOOAX OXXOXXOAA* OAXOXXOA* OAXOXXOAX*
OAAXXOAAA OAOXXOXAA OXOXXOXAO* OAOXXOXXO* OAOXXOXXO OOAXXOAXA OOOXXOXXA* OOOXXOAXX* OAAXXOAOX OAXXXOOOX
OAAOXXAAA OXAOXXOAA* OAOOXXXAA OOOXXXXAA* OOOXXXXAX* OAAOXXOAX* OAOXXAAXA* OOOXXXAAA*
OOOAXXAAX* OOOAXAXAX* OOAAXAAXA OOXAXAAXA OOXOXXOXA* OOXOXXOXX* OOOXXAAXA* OOOAXXAXA* OOOAXXAAXX*
OOOAXXAAX* OOOXXAAAX*

States yang digunakan pada DFA saat user bermain pertama.

```
OXXXXOOOX# OXXAXXOOO* OXXXXAOOO* OXOXXOXXX# OXOOXXXXXX# OXXOXXOOA* OXXOXXOOX* OXXXXOOXX# OXXOXXAOAA* OAXOXXOAA*
OAXOXXOAX* OAXOXXAAA* OXAOXXAAA* OXAOXXAAAA* OXAOXXAAAA* OXAOXXAAAA* OXAOXXAAAA* OXAOXXAAAA* OXAOXXAAAA*
```

```
ΧΑΑΧΧΑΟΑΟ ΑΧΑΑΧΑΑΟΟ ΑΑΧΑΧΑΟΑΟ ΑΑΑΧΧΟΑΑΟ ΑΑΑΑΧΑΑΑΟ ΑΑΑΟΧΧΑΑΟ ΑΑΟΑΧΑΑΟ ΑΟΑΧΑΑΧΟ ΑΑΑΑΧΑΑΑΟ
OXXAXAOOA OXXAXAOOA OXXAXAOOA OXXAXAOOA OXXAXAOOA OXXAXXOOO* OXXAXAOOA OXXAXAOOA OXXAXAOOA OXXAXAOOA OXXAXAOOA
OXXAXOOX OXXAXOOX OXXAXOOX OXXAXOOX OXXXXXOX # OXXAXOOX OXXAXOOX OXXAXOOX OXXAXOOX OXXAXOOX
0XXXX000X# 0XXXX000X# 0XXXX000X# 0XXXX000X# 0XXXX000X# 0XXXX000X# 0XXXX000X# 0XXXX000X# 0XXXX000X# 0XXXX000X#
0XXAXX000* 0XXAXX000* 0XXAXX000* 0XXAXX000* 0XXAXX000* 0XXAXX000* 0XXAXX000* 0XXAXX000* 0XXAXX000* 0XXAXX000*
0XXXXA000* 0XXXXA000* 0XXXXA000* 0XXXXA000* 0XXXXA000* 0XXXXA000* 0XXXXA000* 0XXXXA000* 0XXXXA000*
XXXXXXX AOAOXXXXX AOAOXXXXXX AOAOXXXXX AOAOXXXX AOAOXXXXX AOAOXXXX AOAOXXX AOAXXX A
#XXXXXXXX ACOCXXXXX ACOCXXXX ACOCXXX ACOCXX ACOCXXX ACOCXXX ACOCXXX ACOCXXX ACOCXXX ACOCXXX ACOCXXX ACOCXXX A
OXOXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA
OXOXXOAOX OXOXXOAOX OXOXXXOAOX OXOXXOAOX OXOXXOAOX OXOXXOAOX OXOXXXOXO# OXOXXOAOX OXOXXOAOX
OXAOXXAOA OXAOXXAOA OXAOXXAOA OXXOXXXOOA* OXAOXXAOA OXAOXXAOA OXOXXXXAOA OXAOXXXAOA OXAOXXXAOA OXAOXXXAOA OXAOXXAOA
OXOOXXXOX# OXOOXXXOX# OXOOXXXOX# OXOOXXXOX# OXOOXXXOX# OXOOXXXOX# OXOOXXXOX# OXOOXXXOX# OXOOXXXOX# OXOOXXXOX#
OXXOXXOOA* OXXOXXOOA* OXXOXXOOA* OXXOXXOOA* OXXOXXOOA* OXXOXXOOA* OXXOXXOOA* OXXOXXOOA* OXXOXXOOA*
OXAOXXOOX* OXAOXXOOX* OXAOXXOOX* OXAOXXOOX* OXAOXXOOX* OXAOXXOOX* OXAOXXOOX* OXAOXXOOX* OXAOXXOOX*
XOXAXOOXO XOXAXOOXO XOXAXOOXO XOXAXOOXO XOXAXOOXO XOXAXOOXO XOXAXOOXO XOXAXOOXO XOXAXOOXO
XOAAXAOXO XOAAXAOXO XOXAXOOXO XOAXXOOXO XOAAXAOXO XOAAXAOXO XOAAXAOXO XOAAXAOXO XOAAXAOXO
XOAXXOOXO XOAXXOOXO #XOXXXOOXO XOAXXOOXO XOAXX
OAXAXAOAA OAXAXAOAA OXXOXAOAA* OAXAXAOAA OAXXXOOAA OAXAXAOAA OAXOXXOAA* OAXAXAOAA OAXOXAOXA* OAXOXAOAX*
ACOCXXXOO AACOCXXXOO AACOCXXXAO AACOCXXXAO AACOCXXXAO AACOCXXXOO AACOCXXXOO AACOCXXXOO AACOCXXXOO AACOCXXXOO AACOCXXXAO AACOCXXXAO AACOCXXXAO AACOCXXXAO AACOCXXXAO AACOCXXXAO AACOCXXXAO AACOCXXXAO AACOCXXAO AACOCXXAO
#XXOOXXXOO AXOOXXXOO AXOOXXXOO AXOOXXXOO AXOOXXXOO AXOOXXXOO AXOOXXXOO AXOOXXXOO AXOOXXXOO
OOXXXOOXX# OOXXXOOXX# OOXXXOOXX# OOXXXOOXX# OOXXXOOXX# OOXXXOOXX# OOXXXOOXX# OOXXXOOXX# OOXXXOOXX#
OOXXXOOAX OOXXXOOAX OOXXXOOAX OOXXXOOAX OOXXXOOAX OOXXXOOAX OOXXXOOAX OOXXXOOAX OOXXXOOAX
OXXOXAOAA* OXXOXAOAA* OXXOXAOAA* OXXOXAOAA* OXXOXAOAA* OXXOXAOAA* OXXOXAOAA* OXXOXAOAA* OXXOXAOAA*
OAXOXXOAA* OAXOXXOAA* OAXOXXOAA* OAXOXXOAA* OAXOXXOAA* OAXOXXOAA* OAXOXXOAA* OAXOXXOAA* OAXOXXOAA*
OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA*
OAXOXAOAX* OAXOXAOAX* OAXOXAOAX* OAXOXAOAX* OAXOXAOAX* OAXOXAOAX* OAXOXAOAX* OAXOXAOAX* OAXOXAOAX*
XOAOXXAAO AAXOXXAOO AAXOXXOAO AAAOXXAAO AAAOXXAAO AAAOXXAAO AAOXXAAO AAAOXXAAO AAAOXXAAO AAAOXXAAO
OAOXXOXAA OAOXXOXAAO OXOXXOXAO* OAOXXOXAA OAOXXOXAA OAOXXOXAA OAOXXOXAA OAOXXOXXO* OAOXXOXXO
OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO*
OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO*
OAOXXOXOX OAOXXOXOX OXOXXOXOX# OAOXXOXOX OAOXXOXOA OAOXXOXOX OAOXXOXOX OAOXXOXOX OAOXXOXOX
XXAOXXOOO AXAOXXAOO AXXOXXOOO AXAOXXAOO AXAOXXAOO AXAOXXAOO AXOOXXXOO AXAOXXAOO AXAOXXAOO
000XX0XXA* 000XX0XXA* 000XX0XXA* 000XX0XXA* 000XX0XXA* 000XX0XXA* 000XX0XXA* 000XX0XXA* 000XX0XXA* 000XX0XXA*
OOOXXOAXX* OOOXXOAXX* OOOXXOAXX* OOOXXOAXX* OOOXXOAXX* OOOXXOAXX* OOOXXOAXX* OOOXXOAXX* OOOXXOAXX*
XOAOXXAAO XOAOXXAAO XOXOXXOAO XOAOXXAAO XOAOXXAAO XOAOXXAAO XOOOXXXAA XOAOXXAAO XOAOXXAAO
COOCXXXAO XOOOXXXAO XOOOXXXAO XOOOXXXAO XOOOXXXAO XOOOXXXAO #XOOOXXXXO XOOOXXXAO XOOOXXXAO
OAAOXXAAA OAAOXXAAA OXAOXXOAA* OAXOXXXAA* OAAOXXAAA OAAOXXAAA OAAOXXXAAA OAOXXXAA OAAOXXOXA* OAAOXXXAXX
*AAOXXOAA* OXAOXXOAA* OXAOXXOAA* OXAOXXOAA* OXAOXXOAA* OXAOXXOAA* OXAOXXOAA* OXAOXXOAA* OXAOXXOAA*
*XXXXXOOOO *XXXXXOOOO AAXXXOOAO AAXXXOOAO AAXXXOOAO AAXXXOOAO AAXXXOOAO AOXXXXOOAO
0000XXXXA* 0000XXXXA* 0000XXXXA* 0000XXXXA* 0000XXXXA* 0000XXXXA* 0000XXXXA* 0000XXXXA* 0000XXXXA*
0000XXXAX* 0000XXXAX* 0000XXXAX* 0000XXXAX* 0000XXXAX* 0000XXXAX* 0000XXXAX* 0000XXXAX* 0000XXXAX*
ACOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA*
OAAOXXOAX* OAAOXXOAX* OAAOXXOAX* OAAOXXOAX* OAAOXXOAX* OAAOXXOAX* OAAOXXOAX* OAAOXXOAX* OAAOXXOAX*
OADAXAXAA DADAXAXAA OXOAXAXOA DADAXAXAA OOXXAXAA* DADAXAXAA OOXXXXAA* DADAXAXAA OOXXXXXX* OOXXXXXXX
000XXAXAA* 000XXAXAA* 000XXAXAA* 000XXAXAA* 000XXAXAA* 000XXAXAA* 000XXAXAA* 000XXAXAA* 000XXAXAA*
*AXXXAAP* 000AXXXAA* 000AXXXAAP* 000AXXXAAP* 000AXXXAAP* 000AXXXAAP* 000AXXXAAP* 000AXXXAAP* 000AXXXAAP* 000AXXXAAP*
OODAXAXXA* OOOAXAXXA* OOOAXAXXA* OOOAXAXXA* OOOAXAXXA* OOOAXAXXA* OOOAXAXXA* OOOAXAXXA* OOOAXAXXA*
OOOAXAXAX* OOOAXAXAX* OOOAXAXAX* OOOAXAXAX* OOOAXAXAX* OOOAXAXAX* OOOAXAXAX* OOOAXAXAX* OOOAXAXAX*
ODAAXAAXA ODAAXAAXA OOAAXAAXA OOXAXAOXA OOXXXAAXA* OOAAXAAXA OOOAXXAXXA* OOAAXAAXAA OOOAXAAXXX
*XXOAXOXO AXOAXAXOO AXOAXAXOO *XAOXXOXOO AXOAXAXOO AXOOXXXOO AXOAXAXOO AXOAXAXOO AXOAXAXOO
OXXXXXXA* OXXXXXXA* OXXXXXXA* OXXXXXXA* OXXXXXXA* OXXXXXXA* OXXXXXXA* OXXXXXXA* OXXXXXXA* OXXXXXXA*
OOXOXAOXX* OOXOXAOXX* OOXOXAOXX* OOXOXAOXX* OOXOXAOXX* OOXOXAOXX* OOXOXAOXX* OOXOXAOXX* OOXOXAOXX*
OOOXXAAXA* OOOXXAAXA* OOOXXAAXA* OOOXXAAXA* OOOXXAAXA* OOOXXAAXA* OOOXXAAXA* OOOXXAAXA* OOOXXAAXA*
OODAXXAXA* OODAXXAXA* OODAXXAXA* OODAXXAXA* OODAXXAXA* OODAXXAXA* OODAXXAXA* OODAXXAXA* OODAXXAXA*
OODAXAAXX* 000AXAAXX* 000AXAAXX* 000AXAAXX* 00AXAAXX* 00AXAAXX* 00AXAAXX* 00AXAAXX* 00AXAAXX* 00AXAAXX
OAOAXAAAX OXOAXAAOX OXOAXAAOX OAOAXAAAX OOOXXAAAX* OAOAXAAAX OOOAXAAXX* OAOAXAAXX* OAOAXAAAX
OODAXXAAX* OODAXXAAX* OOOAXXAAX* OOOAXXAAX* OOOAXXAAX* OOOAXXAAX* OOOAXXAAX* OOOAXXAAX* OOOAXXAAX*
OOOXXAAAX* OOOXXAAAX* OOOXXAAAX* OOOXXAAAX* OOOXXAAAX* OOOXXAAAX* OOOXXAAAX* OOOXXAAAX* OOOXXAAAX*
```

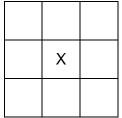
Transition table pada DFA saat user bermain pertama

Contoh cara membaca *transition table*:

Misal *states* sekarang adalah **OAAAXAAA**(*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.



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 **OAAAXAAA**.

0		
	X	

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

0	Χ	
	Х	
	0	

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* **OXAAXAOA.** Setelah itu *user* akan kembali diminta meng*input* posisi untuk simbol 'X' ditempatkan. Kemudian kali ini *user menginput* 3. Maka *states* akan berpindah menuju *states* **OXXAXAOOA**

0	Χ	Χ
	Χ	
0	0	

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.

0	X	X
Χ	X	
0	0	0

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:

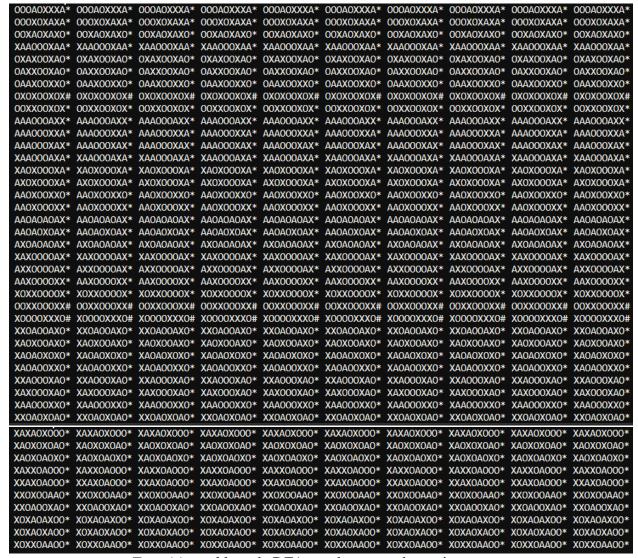
- 1. Alphabet yang dipakai {1, 2, 3, 4, 5, 6, 7, 8, 9}
- 2. Komputer akan mengisi papan dengan posisi nomor 5 terlebih dahulu. Sehingga *start state* saat komputer bermain pertama adalah **AAAOAAA**
- 3. *Final states* saat komputer bermain pertama ada 77 *states*. *State* ini akan menghasilkan permainan 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.

States yang digunakan pada DFA saat komputer bermain pertama

Final states pada DFA saat komputer bermain pertama

XAOAOAAAA AXAOOAAAA AAXOOAAAA AAAOAAOA AAAAOAAAA AAAAOXOAA AAAAOOXAA AAAAOOAXA OAAAOOAXA AAAAOAAAA OAXAOOAAX *OXXOOOAX OAXAOOAX OAXXOOAOX OAXAOOAAX OAXAOOAAX *OAXOOOAX *OAXOOOAX OAXAOOAAX OAXAOOA XXX00AOX #OXXX000OX OAXX00AOX OAXX00AOX OAXX00AOX *00XX00XOX OAXX00AOX OAXX00AOX OAXX00AOX OAXX00AOX OAOXOAAAX *OXOXOAOAX OAOXOAAAX OAOXOAAAX OAOXOAAAX OAOXOAAAX OAOXOAAAX OAOXOAAAX XAAXOAOAO XAAXOAOOO XXAXOAOOO* XAXXOAOOO* XAAXOAOAO XAAXOAOAO XAOXOXOAO* XAAXOAOAO XAOXOAOXO* XAAXOAOAO OAAAOAOXX *OXAOOAOXX *OAXOOAOXX *OAAOAOXX OAAAOAOXX OAAAOAOXX OAAAOAOXX OAAAOAOXX OAAAOAOXX OAAAOAOXX XOXOOAXO XOXOOAXO XOXOOAXO XOXOOAXO XOXOOAXO XOXOOAXO XOXOOXOXO* XOXOOXOXO* XOXOOXOXO XOXOOXOXO* AAXOOAAAA XAXOOOAAA* AXXOOOAAA* AAXOOAAAA AAXOOAAAA AAXOOAAAA AAXOOAAAA AAXOOAAAA AAXOOAAA OAXOOXAA *OXAXOOXAO *OAXXOOXAO OAAXOOXAA OAAXOOXAA OAAXOOXAA OAAXOOXAA *OAAXOOXAO OAAXOOXAO OAAXOOXAA OAAXOOXOX *OOXXOOXOX *OOXXOOXOX OAAXOOXOX OAAXOOX OAAXOOXOX OAAXOOXOX OAAXOOX OAAX AAOXOAAA XAOXOAOAA* AXOXOAOAA* AAOXOAAA AAOXOAAA AAOXOAAA AAOXOAAA AAOXOAAA AAOXOAAA AAOXOAAA* A OAOXOAXAA OAOXOAXAA OXOXOAXAA* OAOXOAXAA OAOXOAXAA OAOXOAXAA OAOXOAXAA* OAOXOAXAA* OAOXOAXAA* OAOXOAXAA* ACAACAACA ACAACAACA *ACAXCAACA *A OCAOXAXA OOAOXXX OOAOXXX OOAOXXX OOXOXXXX OOAOXXX OOAXXX *XAXOOOAAA *AXXOOOAAA AAXOOAAAA AAXOOAAAA AAXOOAAAA AAXOOXAA *AAXOOOAAA *AAXOOOAAA AAXOOOAAA AAXOOAAA AAXOOAAA OAAXOOXAA OAAXOOXAA OXAXOOXAO* OAXXOOXAO* OAAXOOXAA OAAXOOXAA OAAXOOXAA OAAXOOXOX OAAXOOXXO* OAAXOOXOX OAAXOOXOX OXOXOOXOX# OOXXOOXOX* OAAXOOXOX OAAXOOXOX OAAXOOXOX OAAXOOXOX OAAXOOXOX OAAXOOXOX *XXAOOOAAA *AXAOOAAA *AXXOOOAAA AXAOOAAAA AXAOOAAAA AXAOOOAAA *AXAOOOAAA *AXAOOOAAA *AXAOOOAAA *AXAOOAAA * AOXXOOOXX XOXXOOOXX* AOXXOOOAX AOXXOOOAX AOXXOOOAX AOXXOOOAX AOXXOOOAX AOXXOOOAX *AAOAOXOAX *AAOAOAOXOAX *AAOAOXOAX *AAOAOXOAX *AAOAOXOAX *AAOAOXOAX *AAOAOXOAX *AAOAOXOXAX *AAOAOXOXAX *AAOAOXOXAX XAOAOAOAX* AXX000AAA* AXX000AAA* AXX000AAA* AXX000AAA* AXX000AAA* AXX000AAA* AXX000AAA* AXX000AAA* AXX00AAA* *AAX000XAA* AAX000XAA* AAX000XAA* AAX000XAA* AAX000XAA* AAX000XAA* AAX000XAA* AAX000XAA* AAX000XAA *AAXOOOAA* AXAOOOAA* AXAOOOAA* AXAOOOAA* AXAOOOAA* AXAOOOAA* AXAOOOAA* AXAOOOAA* AXAOOOAA* AXAOOOAA* AXAOOOAXA* AXAOOOAXA AXAOOOAX* AXAOOOAX* AXAOOOAX* AXAOOOAX* AXAOOOAX* AXAOOOAX* AXAOOAX* AXAOOAX* AXAOOOAX* AXAOOOAX* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* OXAOOXXOAO* OXXOOAXO* OXAOOXXOAO* OXXOOAXO* OXXOOAXO* OXXOOAXO* OXXOOAXO* OXXOOAXO* OXXOOAXO* OXAOOXXOAO* OXAOOXXOAO* 0XA00XAX0* OXAOOXOAX* OXAOOXOAX* OXAOOXOAX* OXAOOXOAX* OXAOOXOAX* OXAOOXOAX* OXAOOXOAX* OXAOOXOAX* OXAOOXOAX* AAXOOOAAX* AAXOOOAAX* AAXOOOAAX* AAXOOOAAX* AAXOOOAAX* AAXOOOAAX* AAXOOOAAX* AAXOOOAX* AAXOOOAX* AAXOOOAXA* AAXOOOAXA* AAXOOOAXA* AAXOOOAXA* AAXOOOAXA* AAXOOOAXA* AAXOOOAXA* AAXOOOAXA* AAXOOOAXA* XAX0000AA* XAX000AAA* XAX000AAA* XAX000AAA* XAX000AAA* XAX00AAA* XAX00AAA* XAX00AAA* XAX00AAA* XAX0AAA* XAX0AAA OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXAXO* OAXOOXAXO* OAXOOXAXO* OAXOOXAXO* OAXOOXAXO* OAXOOXAXO* OAXOOXAXO* OAXOOXAXO* OAXOOXAXO* X0X00X0X0# X0X00X0# X0X00XX00* AXOXOAOAA* AXOXOAOAAA* AXOXOAOAAAA AXOXOAOAAAA AXOXOAOAAAA AXOXOAOAAAA AXOXOAOAAAAAA AXOXOAOAAAA AXOXOAOAAAA AXOXOAOAAAAAAA AXOXOAOAAAAA AXOXOA AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOAOXA* AAOXOAOAX* AAOXOAOAX* AAOXOAOXX* AAOXX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOXOAXXA* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* 000X0XXAA* 000X0XXAA* 000X0XXAA* 000X0XXAA* 000X0XXAA* 000X0XXAA* 000X0XXAA* 000X0XXAA* 000X0XXAA* OXOXOAXAO* *AOAXOAAOX *AOAXOAAOX *AOAXOAAOX AAOAXOAAOX *AOAXOAAOX *AOAXOAAOX *AOAXOAAOX AAOAXOAAOX *AOAXOAAOX *AOAXOAAOX AOXAOXAOA* AOAXOXAOA* *AOXXOAAOA *XOAXOAAOA *XOAXOAAOA *XOAXOAAOA *XOAXOAAOA *XOAXOAAOA *XOAXOAAOA *XOAXOAAOA *XOAXOAAOA *XOAXOAAOA *XOAXOAAOA

OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX*



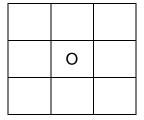
Transition table pada DFA saat komputer bermain pertama

Contoh cara membaca transition table di atas:

Misalkan *states* sekarang adalah **AAAAOAAA** (*start state*), lalu *user* menginput 2, maka *states* tersebut akan berpindah menuju *states* **AXAOOAAA**

Contoh masukan pada DFA *user* bermain pertama:

1. Komputer akan bermain pertama dan langsung menempatkan simbol 'O' ke posisi 5 pada papan. Oleh karena itu, *start state* dari DFA ini adalah **AAAOAAA**



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

	Х	
0	0	

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**.

0	Х	
0	0	Х

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

0	Х	Х
0	0	Х
		0

- 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*.

BAB III

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):

```
AAXOOAAAA AXAXOAAOO AAAXOOAOA AAXAOXOAO AAAAOXOAA OAAXOOXOX OAAXOOXAA AAAAOOXAA AAAXOOAXO AAAAOOAXA
*AAAOOOAXX *XAOAOAAAA AXAOAOXOAX XAOOOXXAA XAOOOAXAA XAOAOAAAA AXAOOXAAA AXAOOXAAA AXAOOAAAA AXAOAOAAA *
AAXOOOAXA* XAXOOOAAA* OAXOOXXAO* OAXOOXAXO* XOXOOXXOO# XOXOOXXOO* AXOXOAAA* AAOXOAOAA* AAOXOAOAA* AAOXOAOAA*
000A0XXXA* 000X0XAXA* 00XX0XAXO* XAA000XAA* 0XXX0XAO* 0AXX0XAO* 0AXX0XXXO* 0XXX0XXX# 0XXX0XXXX AAA000XXX*
AAAOOOXXA* AAAOOOXAX* XAAOOOAXA* XAOXOOXA* AXOXOOXA* AAOXOOXXO* AAOXOOXXX* AAOAOAOAX* AAOAOXAX* AXOAOAX*
XXAOOOXAO* XAXOOOXAO* XAAOOOXXO* XXOAOXOAO* XAXAOXOOO* XAOXOAOXO* XAXXOAOOO* XXAXOAOOO* XXOXOOAAO*
0XXXXA000* 0XAXX0A0A 0XXXX000A 0XOXX0XOA 0XOXX0XOX# 0XXXXA0A 0XA0XXA0A 0X00XXXXA 0X00XXXXX# 0XXXXXA0A*
000XX0XA* 000XX0AXX* 0AAXX0A0X 0AXXX00X 0AA0XXAAA 0XA0XXAA* 0A0XXXAA* 000XXXAX* 0AA0XXXAX* 0AA0XXXAX*
OAAOXXOAX* OAOAXAXAA OOOXXAXAA* OOOAXXXAA* OOOAXAXXAX OOOAXAXAXX OOAAXAAXA OOXAXAXAA OOXAXAXA OOXAXAXA OOXAXAXA
OOOXXAAXA* OOOAXXAXA* OOOAXAAXX* OAOAXAAAX OOOAXXAAX* OOOXXAAAX*
InitialStates:
ΑΑΑΑΟΑΑΑ ΟΑΑΑΧΑΑΑΑ
FinalStates:
0XA00XAX0* 0XA00X0AX* AAX000AAX* AAX000AAX* XAX000AAA* 0AX00XXA0* 0AX00XXX0* X0X00XX00# X0X00XX00* AX0X0AAA*
0XXX0XXXX AAA000AXX* AAA000AXX* AAA000XXX* AAA000XXX* XAA00AXX* XAAXXXX AAAXXXX AAAXXXXX AAAXXXXX AAAXXXXX AAAXXXXX AAAXXXXX AAAXXXXX AAAXXXX AAAXXX AAAXXX AAXXXX AAXXXX AAXXXX AAXXXX AAXXXX AAXXXX AAXXXX AAXXXX AAXXX AXXX 
XAXXOAOO* XXAXOAOO* XXOXOOO* XXOXOAOO* XXOAOOXOO* XXXAXOOO* XXXXXOOOX# OXXXXXXOO * OXXXXXXOOO* OXXXXXXOOO*
OXOXXOXOX# OXOOXXXOX# OXXOXXOOA* OXAOXXOOX* OOXXXOOXX# OXXOXAOA* OAXOXXOAA* OAXOXAOXA* OAXOXAOAX* OXOOXXOXAO*
OAOXXOXXO* OOOXXOXXA* OOOXXOAXX* OXAOXXOAA* OOOXXXXA* OAOXXOXA* OAAOXXOAX* OAOXXOAX* OAOXXOAX* OAOXXOAX*
```

Daftar states, Initial/start states, Final states

DaftarSimbol: 1 2 3 4 5 6 7 8 9

TransitionsTable:

XAOAOAAAA AXAOOAAAA AAXOOAAAA AAAXOAAOA AAAAOAAAA AAAAOXOAA AAAAOOXAA AAAAOOXAA AAAAOOAXA AAAAOAAAX AAAAOAAAA OXAAOAOAX OXAAOAOAX *OXXOOAOAX *OXOXOAOAX OXAAOAOAX *OXAOOXOAX OXAAOAOAX *OXAOOAOXX OXAAOAAAX OXAAOAOAX XAAOOAXAO XAAOOAXAO XXXOOOAAO* XAXOOOXAO* XAAOOAXAO XAAOOAXAO XAAOOAXAO XAAOOAXAO XAAOOAXAO XAAOOAXAO OAXXOOAOX 0AXXOOAOX OAXXOOAOX OAXXOOAOX OAXXOOAOX OAXXOOAOX *OOXXOOXOX OAXXOOAOX OAXXOOAOX OAOXOAAAX *OXOXOAOAX OAOXOAAAX OAOXOAAAX OAOXOAAAX *OAOXOXOAX *OAOXOAOAX OAOXOAOAX OAOXOAAAX OAOXOAAAX OROROXARX *OXOROXORX OROROXARX *OROXOXORX *OROROXARX *OROROXXXX *OXOROXOXX X OROROXARX OROROXARX OAAAOAXOX OXAAOOXOX *OOXAOAXOX *OOAXOAXOX OAAAOAXOX *OOAAOXXOX OAAAOAXOX OAAAOAXOX OAAAOAXOX *XAAOOOAXA *AXAOOOAXA *AXAOOOAXA *AXAOOAXA AAAAOOAXA AAAAOOAXA *AAAOOOXAX *AXAOOOAXA *AXAOOOAXAOOOAXA *AXAOOOAXA *AXAOOOAXA *AXAOOOAXAOOOAXA *AXAOOOAXAOOOAXAOOOAXA *AXAOOOAXAOOOAXAOOOAXA AXAOOOAAA* AXXOOOAA* AAXOOOXAA *AAXOOAAA AAAAOOXAA AAAAOOXAA AAAAOOXAA AAAAOOXAA *AAAOOOXAA AAAAOOXAA AAAAOOXAA OAAXOOXAA *OAXXOOXAO *OAXXOOXAO OAAXOOXAA OAAXOOXAA OAAXOOXAA AOAXOOXAA OAAXOOXAA OAAXOOXAA OAAXOOXOX #OXXOXXOX *OXXXOOXOX OAAXOOXOX OAAXOOXOX OAAXOOXOX OAAXOOXOX #OXOOXOX *OXXXOOXOX OAAXOOXOX OAAXOOXOX #OXOOXOX *OXXXOOXOX *OXXXOOXOX OAAXOOXOX OAAXOOXOX #OXXXOOXOX *OXXXOOXOX OAAXOOXOX OAAXOOXOX OAAXOOXOX OAAXOOXOX OAAXOOXOX #OXXXOOXOX OAAXOOXOX OAAXOOXX OAAXOOXOX OAAXOOX OAAXOOXOX OAAXOOX OA *XOAXOAAOA AXAXOAAOO *AOXXOAAOA AAAXOAAOA AAAXOAAOA *AOAXOXAOA *AOAXOAXOA AAAXOAAOA *AOAXOAAOA AAAXOAAOA *AOAXOAAOA *XXAXOAOOO AXAXOAOOO AXXXOAOOO AXAXOAAOO AXAXOAAOO AXAXOAOOO AXAXOAOOO AXAXOAAOO AXAXOAAOO AXAXOAAOO *XAXOOOAAA *AXXOOOAAA AAXOOAAAA AAXOOAAAA AAXOOXAAO *AAXOOXAAO AAXOOAAA AAXOOAAA AAXOOOAAA AAXOOAAA AAXOOAAAA XXXOXAAO *OXXOXAAO AAXOOXAAO AAXOOXAAO AAXOOXAAO AAXOOXAAO *OAXOOXXAO *OAXOOXAXO AAXOOXAAO AAXOOXAAO OAAXOOXOX OAAXOOXOX OXOXOOXA# OOXXOXOX* OAAXOOXOX OAAXOOXOX OAAXOOXOX OAAXOOXOX OAAXOOXOX AAOOXOAXA XAOOXOOXA* AXOXOOXA* AAOOXOAXA AAOOXOAXA AAOOXOAXA AAOXOOXXA* AAOOXOAXA AAOXOOXXA* XAOOOAXXA XXXOOOXXA XAOOOAXAA XAOOOAXAA XAOOOXXAA XAOOOXXAA XAOOOAXAA XAOOOOXXA XAOOOOXXA XAOOOAXAA

XAOXOAOAA* *XAOAOAOAX *XAOAOAOAX *XAOAOAOAX *XAOAOAOAX *XAOAOAOAX *XAOAOAOAX *XAOAOAOAX *XAOAOAOAX *XAOAOAOAX *XAOAOAAX *XAOAAAX *XAOAOAAX *XAOAOAAX *XAOAOAAX *XAOAAAX *XAOAOAAX *XAOAOAAX *XAOAAAAX *XAOAAAAX *XAOAAAAX *XAOAAAAX *XAOAAAAX *XAOAAAAX *XAOAAAAX *XAOAAAX *XAOAAAAX *XAOAAAX *XAOAAX *XAOAAAX *XAOAAAX *XAOAAAX *XAOAAAX *XAOAAAX *XAOAAAX *XAOAAAX *XAOAAX *XAOAAX *XAOAAX *XAOAAX *XAOAAX *XAOAAX *XAOAAAX *XAOAAX *XA *AAA0000AXX *AAA0000AXX *AAA0000AXX *AAA0000AXX *AAA0000AXX *AAA0000AXX *AAA0000AXX *AAA0000AXX AXXOOOAA* AXXOOOAA* AXXOOOAA* AXXOOOAA* AXXOOOAA* AXXOOOAA* AXXOOOAA* AXXOOOAA* AAXOOOXAA* AAXOOOXAA* AAXOOOXAA* AAXOOOXAA* AAXOOOXAA* AAXOOXAA* AAXOOXAA* AAXOOXAA* AAXOOXAA* AAXOOXAA* AAXOOXAA* *AXAOOOAXA AXAOOOAX* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* 0XX00XAA0* OXAOOXXAO* OXAOOXXAO* OXAOXXOAO* OXAOOXXAO* OXAOOXXAO* OXAOOXXAO* OXAOOXXAO* OXAOOXXAO* OXAOOXXAO* OXAOOXXAO* *OXAOOAXO *OXAOOAXO OXAOOAXO OXAOOAXO OXAOOAXO OXAOOAXO OXAOOAXO OXAOOAXO OXAOOAXO OXAOOAXO OXAOOAXO 0XA00X0AX* 0XA00X0AX* 0XA00X0AX* 0XA0X0AX* 0XA0X0AX* 0XA0X0AX* 0XA0X0AX* 0XA0X0AX* 0XA0X0AX* 0XA0X0AX* AAXOOOAAX* AAXOOOAAX* AAXOOOAAX* AAXOOOAAX* AAXOOOAXX AAXOOOAXX* AAXOOOAXX* AAXOOOAXX* AAXOOOAXX* AAX000AXA* AAX000AXA* AAX000AXA* AAX000AXA* AAX000AXA* AAX000AXA* AAX000AXA* AAX000AXA* AAX000AXA* XAX000AAA* XAX000AAA* XAX000AAA* XAX000AAA* XAX000AAA* XAX000AAA* XAX000AAA* XAX000AAA* XAX000AAA* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXXAO* OAXOOXAXO* AXOXOAOAX* AXOXOAOAA* AXOXOAOAX* AXOXOAOAA* AXOXOAOAX* AXOAOAOXOXA* AXOXOAOAA* AXOXOAOAA* AXOXOAOAX* AXOXOAOAX AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA* AAOXOXOAA AAOXOAOXA* AAOXOAOAX* AAOXOAOXX* AAOXOAOXX* AAOXOAOXX* AAOXOAOXX* AAOXOAOXX* AAOXOAOXX* AAOXOAOXX* AAOXOAOXX* AAOXOAOXX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXAX* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* OOOXOAXXA* 000X0XXAA* OXOXOAXAO* OXOXOAXAO* OXOXOAXAO* OXOXOAXAO* OXOXOAXAO* OXOXOAXAO* OXOXOAXAO* OXOXOAXAO* OXOXOAXAO* *AOAXOAAOX *AOAXOAAOX *AOAXOAAOX *AOAXOAAOX *AOAXOAAOX *AOAXOAAOX *AOAXOAAOX *AOAXOAAOX *AOAXOAAOX *AOAXOXAOA* AOAXOXAOA* AOAXOXAOA* AOAXOXAOA* AOAXOXAOA* AOAXOXAOA* AOAXOXAOA* AOAXOXAOA* AOAXOXAOA* AOAXOXAOA AOAAOXXOA* AOAAOXXOA* AOAAOXXOA* AOAAOXXOA* AOAAOXXOA* AOAAOXXOA* AOAAOXXOA* AOAAOXXOA* AOAAOXXOA* AOAAOXXOA*

```
AOAAOXAOX* AOAAOXAOX* AOAAOXAOX* AOAAOXAOX* AOAAOXAOX* AOAAOXAOX* AOAAOXAOX* AOAAOXAOX* AOAAOXAOX* AOAAOXAOX*
OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX* OOOAOXAXX*
000A0XXXA* 000A0XXXA* 000A0XXXA* 000A0XXXA* 000A0XXXA* 000A0XXXA* 000A0XXXA* 000A0XXXA* 000A0XXXA*
000X0XAXA* 000X0XAXA* 000X0XAXA* 000X0XAXA* 000X0XAXA* 000X0XAXA* 000X0XAXA* 000X0XAXA* 000X0XAXA*
00XA0XAX0* 
OXAXOOXAO* OXAXOOXAO* OXAXOOXAO* OXAXOOXAO* OXAXOOXAO* OXAXOOXAO* OXAXOOXAO* OXAXOOXAO* OXAXOOXAO*
OAXXOXAO* OAXXOXAO* OAXXOXAO* OAXXOXAO* OAXXOXAO* OAXXOXAO* OAXXOXAO* OAXXOXAO* OAXXOXAO* OAXXOXAO*
0AAX00XX0* 0AAX00XX0* 0AAX00XX0* 0AAX00XX0* 0AAX00XX0* 0AAX00XX0* 0AAX00XX0* 0AAX00XX0* 0AAX00XX0* 0AAX00XX0*
0X0X00X0X# 0X0X00X0X#
OOXXOOXOX* OOXXOOXOX* OOXXOOXOX* OOXXOOXOX* OOXXOOXOX* OOXXOOXOX* OOXXOOXOX* OOXXOOXOX* OOXXOOXOX*
AAAOOOAXX* AAAOOOAXX* AAAOOOAXX* AAAOOOAXX* AAAOOOAXX* AAAOOOAXX* AAAOOOAXX* AAAOOOAXX* AAAOOOAXX* AAAOOOAXX*
AAAOOOXXA* AAAOOOXXA* AAAOOOXXA* AAAOOXXA* AAAOOXXA* AAAOOXXA* AAAOOXXA* AAAOOXXA* AAAOOXXA* AAAOOXXA*
AAAOOOAAX* AAAOOOAXX* AAAOOOAXX** AAAOOOAXX*** AAAOOOAXX*** AAAOOOAXX*** AAAOOOAXX*** AAAOOOAX
XAAOOOAXA* XAAOOOOAXA* XAAOOOOAXA* XAAOOOOAXA* XAAOOOOAXA* XAAOOOOAXA* XAAOOOOAXA* XAAOOOOAXA* XAAOOOOAXA* XAA
AX0X000XA* AX0X000XA* AX0X000XA* AX0X000XA* AX0X000XA* AX0X00XA* AX0X00XA* AX0X00XA* AX0X00XA* AX0X0XA*
AAOXOOXXO* AAOXOOXXO* AAOXOOXXO* AAOXOOXXO* AAOXOOXXO* AAOXOOXXO* AAOXOOXXO* AAOXOOXXO* AAOXOOXXO* AAOXOOXXO*
AAOXOOOXX* AAOXOOOXX* AAOXOOOXX* AAOXOOOXX* AAOXOOOXX* AAOXOOXX* AAOXOOXX* AAOXOOXX* AAOXOOXX* AAOXOOXX*
AAOAOAOAX* AAOAOAOAX* AAOAOAOAX* AAOAOAOAX* AAOAOAOAX* AAOAOAOAX* AAOAOAOAX* AAOAOAOAX* AAOAOAOAX* AAOAOAOAX*
AAOAOXOAX* AAOAOXOAX* AAOAOXOAX* AAOAOXOAX* AAOAOXOAX* AAOAOXOAX* AAOAOXOAX* AAOAOXOAX* AAOAOXOAX*
AXOAOAOAX* AXOAOAOAX* AXOAOAOAX* AXOAOAOAX* AXOAOAOAX* AXOAOAOAX* AXOAOAOAX* AXOAOAOAX* AXOAOAOAX* AXOAOAOAX*
AXX0000AX* AXX0000AX* AXX0000AX* AXX000AX* AXX000AX* AXX000AX* AXX000AX* AXX000AX* AXX000AX* AXX000AX*
AAX00000XX* AAX00000XX* AAX00000XX* AAX00000XX* AAX0000XX* AAX0000XX* AAX0000XX* AAX0000XX* AAX0000XX*
X0XX0000X* X0XX0000X* X0XX0000X* X0XX0000X* X0XX0000X* X0XX0000X* X0XX0000X* X0XX0000X* X0XX0000X* X0XX0000X*
OOXXOOOXX# OOXXOOOXX# OOXXOOOXX# OOXXOOOXX# OOXXOOOXX# OOXXOOOXX# OOXXOOOXX# OOXXOOOXX# OOXXOOOXX#
X0000XXX0# 
XX0A00AX0* XX0A0AX0* XX
XAOAOXOXO* XAOAOXOXO* XAOAOXOXO* XAOAOXOXO* XAOAOXOXO* XAOAOXOXO* XAOAOXOXO* XAOAOXOXO* XAOAOXOXO* XAOAOXOXO*
XAOAOOXXO* 
XAAOOOXXO* 
*000X0AXAX *000X0AXAX *000X0AXAX *000X0AXAX *000X0AXAX *000X0AXAX *000X0AXAX *000X0AXAX *000X0AXAX
*OAOXOXOAX *OAOXOXOAX *OAOXOXOAX *OAOXOXOXAX *OAOXOXOAX *OAOXOXOAX *OAOXOXOAX *OAOXOXOXAX *OAOXOXOAX *OAOXOXOAX
*OXOAOXOA" XAOXOAOXO* XAOXOAOXO* XAOXOAOXOA* XAOXOAOXOA* XAOXOAOXOA* XAOXOAOXOA* XAOXOAOXOA* XAOXOAOXOA* XAOXOAOXOA*
*000A0XXXX *000A0XXXX *000A0XXAX *000A0XXXX *000A0XXAX *000A0XXXX *000A0XXXX *000A0XXXX *000A0XXXX
**OAAOOXOXX *OAAOOXOXX *OAAOOXOXX *OAAOOXOXX XOAAOOXOXX *OAAOOXXX *OAAOOXXXX *OAAOOXXXX *OAAOOXXXX *OAAOOXXXX
*OAXOOAOXX *OAXOOAOXX *OAXOOAOXX *OAXOOAOXX *OAXOOAOXX *OAXOOAOXX *OAXOOAOXX *OAXOOAOXX OAXOOAOXX *OAXOOAOXX *OAXOOAOXX
*OOXAOAXOO* XOXAOAXOO* XOXAOAXOO* XOXAOAXOO* XOXAOAXOO* XOXAOAXOO* XOXAOAXOO* XOXAOAXOO* XOXAOAXOO*
XAAXAAAA AAAXAAOO AAXAXAOO AAXAXAOAO AAAAXAAAO AAAOXXAAO AAOAXAXAO AOAXAAAO AAAAXAAOO
XOAAXAOXO AOAAXAAXO AOXAXAOXO AOAXXOAXO AOAAXAAXO AOAOXXAXO AOOAXAXXO AOAAXAAXO AOAAXAAXO
OXXXXOOOX# OXXXXOOOX# OXXXXOOOX# OXXXXOOOX# OXXXXOOOX# OXXXXOOOX# OXXXXOOOX# OXXXXOOOX# OXXXXOOOX# OXXXXOOOX#
XXXXXXX AOAOXXXXX AOAOXXXXXX AOAOXXXXX AOAOXXXXX AOAOXXXXX AOAOXXXXX AOAOXXXXX AOAOXXXXX AOAOXXXXXX
#XOOOXXXXO AOOXXXXXO AOOXXXXX AOOXXXX AOOXXXXX AOOXXXXX AOOXXXX AOOXXXX AOOXXXXX AOOXXXX AOOXXX AOOXXX
OXOXXOXOA OXOXXOXOA OXOXXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA OXOXXOXOA
XOAOXXOXO XOAOXXOXO #XOXOXXOXO XOAOXXOXO XOAOXXOXO XOAOXXOXO XOAOXXOXO XOAOXXOXO XOAOXXOXO
*XOOXXOAXO AOAXXOAXO AOXXXOOXO AOAXXOAXO AOAXXOAXO AOAXXOAXO *AOOXXOAXO AOAXXOAXO AOAXXOAXO AOAXXOAXO
OXOOXXXOX# 0XOOXXXOX# 0XOOXXXOX# 0XOOXXXOX# 0XOOXXXOX# 0XOOXXXOX# 0XOOXXXOX# 0XOOXXXOX# 0XOOXXXOX# 0XOOXXXOX#
OXXOXXODA* OXXOXXODA* OXXOXXODA* OXXOXXODA* OXXOXXODA* OXXOXXODA* OXXOXXODA* OXXOXXODA* OXXOXXODA*
XOXAXOOXO XOXAXOOXO XOXAXOOXO #XOXXXXOOXO XOXAXOOXO XOXAXOOXO XOXAXXOOXO XOXAXXOOXO XOXAXXOOXO
XOAAXAOXO XOAAXAOXO XOXAXOOXO XOAXXOOXO XOAAXAOXO XOAOXXOXX XOAAXAOXO XOAAXAOXO XOAAXAOXO XOAAXAOXO
XOXXXOOX XOXXXOOX #XXXXOOXO XOAXXXOOX XOAXXOOX XOAXXOOX XOAXXOOXX XOAXXXOOX XOAXXXOOX XOAXXOOXO XOAXXOOXO
*XAOAXOXAO *AXOAXOXAO AAOAXAXAO AAOAXAXAO AAOAXAXAO AAOOXXXXAO AAOAXAXAO AAOAXOXAO AAOAXOXAO AAOAXAXAO
#XXOOXXXOO AXOOXXXOO AXOOXXXOO AXOOXXXOO AXOOXXXOO AXOOXXXOO AXOOXXXOO AXOOXXXOO AXOOXXXOO
```

```
OOXXXOOXX# OOXXXOOXX# OOXXXOOXX# OOXXXOOXX# OOXXXOOXX# OOXXXOOXX# OOXXXOOXX# OOXXXOOXX# OOXXXOOXX#
OOXXXOOAX OOXXXOOAX OOXXXOOAX OOXXXOOAX OOXXXOOAX OOXXXOOAX OOXXXOOAX OOXXXOOAX
OXXOXAOAA* OXXOXAOAA* OXXOXAOAA* OXXOXAOAA* OXXOXAOAA* OXXOXAOAA* OXXOXAOAA* OXXOXAOAA* OXXOXAOAA*
OAXOXXOAA* OAXOXXOAA* OAXOXXOAA* OAXOXXOAA* OAXOXXOAA* OAXOXXOAA* OAXOXXOAA* OAXOXXOAA* OAXOXXOAA*
OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA* OAXOXAOXA*
OAXOXAOAX* OAXOXAOAX* OAXOXAOAX* OAXOXAOAX* OAXOXAOAX* OAXOXAOAX* OAXOXAOAX* OAXOXAOAX* OAXOXAOAX*
OAAXXOAA OAAXXXOAO AAAXXXOAO AAAXXAAO AAAXXAAO AAAXXAAA AAAXXAAAXXAAA AAAXXAAAXXAAA AAAXXAAAXXAAA AAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAAXXAAXXAAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAXXAAX
OAOXXOXAA OAOXXOXAA OXOXXOXAO* OAOXXOXAA OAOXXOXAA OAOXXOXAA OAOXXOXAA OAOXXOXAA OAOXXOXXO* OAOXXOXOX
OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO* OXOXXOXAO*
OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO* OAOXXOXXO*
OAOXXOXOX OAOXXOXOX OXOXXXXXX# OAOXXOXOX OAOXXOXOX OAOXXOXOX OAOXXOXOX OAOXXOXOX OAOXXOXOX OAOXXOXOX
OOAXXOAXA OOAXXOAXA OOAXXOAXA OOXXXOOXA OOAXXOAXA OOAXXOAXA OOAXXOAXA OOOXXOXXA* OOAXXOAXA OOOXXOAXX*
OOOXXOXXA* OOOXXOXXA* OOOXXOXXA* OOOXXOXXA* OOOXXOXXA* OOOXXOXXA* OOOXXOXXA* OOOXXOXXA* OOOXXOXXA*
000XX0AXX* 000XX0AXX* 000XX0AXX* 000XX0AXX* 000XX0AXX* 000XX0AXX* 000XX0AXX* 000XX0AXX* 000XX0AXX*
OAXXXOOX OAXXXOOX OXXXXOOX# OAXXXOOX OAXXXOOX OAXXXOOX OAXXXOOX OAXXXOOX OAXXXOOX
OAAOXXAAA OAAOXXAAA OXAOXXOAA* OAXOXXOAA* OAAOXXAAA OAAOXXAAA OAAOXXXAA OAAOXXXAA* OAAOXXOXX
OXAOXXOAA* OXAOXXOAA* OXAOXXOAA* OXAOXXOAA* OXAOXXOAA* OXAOXXOAA* OXAOXXOAA* OXAOXXOAA* OXAOXXOAA*
OAOOXXXAA OAOOXXXAA OXOOXXXOA OAOOXXXAA OAOOXXXAA OAOOXXXAA OAOOXXXAA OAOOXXXAA OOOOXXXXA*
0000XXXXA* 0000XXXXA* 0000XXXXA* 0000XXXXA* 0000XXXXA* 0000XXXXA* 0000XXXXA* 0000XXXXA* 0000XXXXA*
0000XXXAX* 0000XXXAX* 0000XXXAX* 0000XXXAX* 0000XXXAX* 0000XXXAX* 0000XXXAX* 0000XXXAX* 0000XXXAX*
OAAOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA* OAAOXXOXA*
OAAOXXOAX* OAAOXXOAX* OAAOXXOAX* OAAOXXOAX* OAAOXXOAX* OAAOXXOAX* OAAOXXOAX* OAAOXXOAX* OAAOXXOAX*
OAOAXAXAA OAOAXAXAA OXOAXAXOA OAOAXAXAA OOOXXAXAA* OAOAXAXAA OOOAXXXXAA* OAOAXAXAA OOOAXAXXXA* OOOAXAXXXX
000XXAXAA* 000XXAXAA* 000XXAXAA* 000XXAXAA* 000XXAXAA* 000XXAXAA* 000XXAXAA* 000XXAXAA* 000XXAXAA*
000AXXXAA* 000AXXXAA* 000AXXXAA* 000AXXXAA* 000AXXXAA* 000AXXXAA* 000AXXXAA* 000AXXXAA* 000AXXXAA*
OOOAXAXXA* OOOAXAXXA* OOOAXAXXA* OOOAXAXXA* OOOAXAXXA* OOOAXAXXA* OOOAXAXXA* OOOAXAXXA*
000AXAXAX* 000AXAXAX* 000AXAXAX* 000AXAXAX* 000AXAXAX* 000AXAXAX* 000AXAXAX* 000AXAXAX* 000AXAXAX*
OOAAXAAXA OOAAXAAXA OOAAXAAXA OOXAXAOXA OOOXXAAXA* OOAAXAAXA OOOAXXAXXA* OOOAXAAXXA OOOAXAAXXX
OOXOXXOXA* OOXOXXOXA* OOXOXXOXA* OOXOXXOXA* OOXOXXOXA* OOXOXXOXA* OOXOXXOXA* OOXOXXOXA* OOXOXXOXA*
OOXOXAOXX* OOXOXAOXX* OOXOXAOXX* OOXOXAOXX* OOXOXAOXX* OOXOXAOXX* OOXOXAOXX* OOXOXAOXX* OOXOXAOXX*
OOOXXAAXA* OOOXXAAXA* OOOXXAAXA* OOOXXAAXA* OOOXXAAXA* OOOXXAAXA* OOOXXAAXA* OOOXXAAXA* OOOXXAAXA*
OOOAXXAXA* OOOAXXAXA* OOOAXXAXA* OOOAXXAXA* OOOAXXAXA* OOOAXXAXA* OOOAXXAXA* OOOAXXAXA* OOOAXXAXA*
OOOAXAAXX* OOOAXAAXX* OOOAXAAXX* OOOAXAAXX* OOOAXAAXX* OOOAXAAXX* OOOAXAAXX* OOOAXAAXX* OOOAXAAXX*
OAOAXAAAX OAOAXAAAX OXOAXAAOX OAOAXAAAX OOOXXAAAX* OAOAXAAAX OOOAXXAAX* OOOAXAAXX* OAOAXAAAX
OOOAXXAAX* OOOAXXAAX* OOOAXXAAX* OOOAXXAAX* OOOAXXAAX* OOOAXXAAX* OOOAXXAAX* OOOAXXAAX* OOOAXXAAX*
OOOXXAAAX* OOOXXAAAX* OOOXXAAAX* OOOXXAAAX* OOOXXAAAX* OOOXXAAAX* OOOXXAAAX* OOOXXAAAX* OOOXXAAAX*
```

B. Kode Program Tic-Tac-Toe

Seperti yang sudah saya jelaskan sebelumnya, program yang saya buat adalah program dalam bahasa C. Nama file untuk program ini adalah "tictactoe.c".

Berikut adalah source code dari program tictactoe.c

```
/*Tic-Tac-Toe Games*/
/* Created on 3 October 2018 by irfansofyana */
#include <stdio.h>
#include <stdbool.h>
#include <stdbool.h>
#include <stdlib.h>

/* Tipe data bentukan untuk membentuk STATES */

typedef struct {
    char States[20];
}STATES;

/*Deklarasi Variabel yang diperlukan program */
char arr[20], now[20], smt[20];
STATES TransitionStates[205][20], DaftarStates[205], FinalStates[120];
STATES InitialStates[5], DaftarSimbol[15];
STATES StatesLewat[20];
```

```
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
bersesuaian */
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;
                        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 */
            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] == '0')
                        printf("|0|");
                  else
                        printf("|X|");
            else if (i%3 == 1){
                  if (arr[i] == 'A')
                        printf(" |");
                  else if (arr[i] == '0')
                        printf("0|");
                  else
                        printf("X|");
            }else{
                  if (arr[i] == 'A')
                        printf(" |\n");
                  else if (arr[i] == '0')
                        printf("0|\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++) {</pre>
            if (strcmp(DaftarStates[i].States, s) == 0) return i;
}
/*Mengecek apakah posisi "pos" pada papan permainan sekarang masih kosong atau
/*Pre-kondosi : 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 \le 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");
            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 \mathbf{x}
                  //akan ditempatkan
                  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");
      //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");
      \{ \text{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 mengulangnya 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 menampilan 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
            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*.