

**APLIKASI PERMAINAN “TIC TAC TOE” DENGAN MENGGUNAKAN FINITE
AUTOMATA
COVER
DOKUMEN LAPORAN**

Disusun untuk memenuhi tugas mata kuliah Teori Bahasa Formal dan Automata

Semester III 2018

Disusun oleh

ABEL STANLEY 13517068




PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2018

I. DESKRIPSI TUGAS

<p style="text-align: center;"><i>Tic-tac-toe</i></p>  <p style="text-align: center;">Penyudah kepada Tic-tac-toe</p>	
Genre	Permainan atas kertas
Pemain	2
Masa penyediaan	Sedikit
Masa permainan	~1 minit
Peluang rawak	Tiada
Kemahiran yang diperlukan	Taktik, strategi dan pemerhatian

Tic-tac-toe (atau dikenali sebagai permainan X dan O) ialah sebuah permainan pensil dan kertas untuk dua pemain, X dan O, yang bergilir-gilir untuk membuat tanda tersebut di dalam sebuah grid 3x3. Biasanya, pemain X yang mulakan permainan. Pemain yang berjaya meletakkan tiga tanda mereka berturut-turut sama ada mendatar, menegak, atau menyerong memenangi permainan ini.

(Wikipedia Bahasa Indonesia)

Pada tugas pertama TBFO ini, kami diminta untuk membuat sebuah permainan tic-tac-toe sederhana, dimana permainan ini akan dimainkan oleh komputer dan player. Program harus bisa dipastikan bahwa komputer **tidak mungkin kalah** didalam permainan. Aplikasi akan membuka file yang berisi informasi mengenai daftar state, daftar simbol, state awal, state akhir, dan transition function. Informasi dari file

tersebut akan digunakan untuk mengecek masukan dari pengguna. Program akan membaca konfigurasi dari file eksternal, dan logika state machine tidak di- *hardcode* ke program secara langsung.

Batasan masalah :

Pada langkah pertama permainan, player/CPU dipastikan meletakkan tanda “X” atau “O” di bagian tengah papan . Telah dipastikan bahwa CPU tidak pernah kalah di dalam permainan ini.

II. NOTASI DFA

1. Transition Table untuk DFA jika pemain gerak duluan

STATES	HUMAN GOES FIRST (X)								
	INPUTS								
	1	2	3	4	5	6	7	8	9
123456789N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	O234X6789N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE
O234X6789N	EEEEEEEEEE	OX34X67O9N	O2X4X6O89N	O23XXO789N	EEEEEEEEEE	O23OXX789N	O2O4X6X89N	OO34X67X9N	O2O4X678XN
OX34X67O9N	EEEEEEEEEE	EEEEEEEEEE	OXX4X6OO9N	OX3XXO7O9N	EEEEEEEEEE	OX3OXX7O9N	OXO4X6XO9N	EEEEEEEEEE	OXO4X67OXN
OXX4X6OO9N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXXXX6OOOC	EEEEEEEEEE	OXXOXXOO9C	EEEEEEEEEE	EEEEEEEEEE	OXXOX6OOXC
OX3XXO7O9N	EEEEEEEEEE	EEEEEEEEEE	OXXXO0O9N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOXXOXO9N	EEEEEEEEEE	OXOXXO7OXN
OXXXO0O9N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXXXO0OXXD
OXOXXOXO9N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOXXOXOXD
OXOXXO7OXN	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOXXOXOXD	EEEEEEEEEE	EEEEEEEEEE
OX3OXX7O9N	EEEEEEEEEE	EEEEEEEEEE	OXXOXXO9C	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOXXOXO9N	EEEEEEEEEE	OX3OXXOXXC
OXOXXOXO9N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOXXOXOXD
OXO4X6XO9N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOXXOXO9N	EEEEEEEEEE	OXOXXOXO9N	EEEEEEEEEE	EEEEEEEEEE	OXOXXOXOXN
OXO4X67OXN	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOXXO7OXN	EEEEEEEEEE	OXOXXO7OXN	OXOXXOXOXN	EEEEEEEEEE	EEEEEEEEEE
OXOXXO7OXN	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOXXOXOXD	EEEEEEEEEE	EEEEEEEEEE
OXOXXOXOXN	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOXXOXOXD	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE
O2X4X6O89N	EEEEEEEEEE	O2XOX6O89C	EEEEEEEEEE	O2XXO089N	EEEEEEEEEE	O2XOXXO89C	EEEEEEEEEE	O2XOX6OX9C	O2XOX6O8XC
O2XXO089N	EEEEEEEEEE	OXXXO0O9N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OOXXO0OX9N	OOXXO08XN
OOXXO0OX9N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OOXXO0XXD
OOXXO08XN	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OOXXO0XXD	EEEEEEEEEE
O23XXO789N	EEEEEEEEEE	OX3XXO7O9N	O2XXO089N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	O2OXXOX89N	OO3XXO7X9N	O23XXO78XN
O2OXXOX89N	EEEEEEEEEE	OXOXXOX8OC	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOXXOX8OC	OOOXXOX8XC
OO3XXO7X9N	EEEEEEEEEE	EEEEEEEEEE	OOXXO0X9N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OOOXXOX9C	EEEEEEEEEE	OOOXXOX9C
OO3XXO78XN	EEEEEEEEEE	EEEEEEEEEE	OOXXO08XN	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OOOXXOX8XC	OOOXXO7XXC	EEEEEEEEEE
O23OXX789N	EEEEEEEEEE	OX3OXXO89C	O2XOXXO89C	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	O2OXXOX89N	O23OXXOX9C	O23OXXO8XC
O2OXXOX89N	EEEEEEEEEE	OXOXXOX89N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OOOXXOX9C	OOOXXOX8XC
O2O4X6X89N	EEEEEEEEEE	OXO4X6XO9N	EEEEEEEEEE	OOOXX6X89C	EEEEEEEEEE	OOO4XX89C	EEEEEEEEEE	OOO4X6X9C	OOO4X6X8XC
OXOXXOXOXN	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOXXOXOXD
OO34X67X9N	EEEEEEEEEE	EEEEEEEEEE	OOX4X6OX9N	OOOXX67X9C	EEEEEEEEEE	OOO4XX7X9C	OOO4X6XX9C	EEEEEEEEEE	OOO4X67XXC
OOX4X6OX9N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OOXXO0X9N	EEEEEEEEEE	OOXOXXOX9C	EEEEEEEEEE	EEEEEEEEEE	OOXOXXOXXC
O2O4X678XN	EEEEEEEEEE	OXO4X67OXN	EEEEEEEEEE	OOOXX678XC	EEEEEEEEEE	OOO4XX78XC	OOO4X6X8XC	OOO4X67XXC	EEEEEEEEEE

List of States :

NORMAL STATES	FINAL STATES (COMPUTER WINS)	FINAL STATES (DRAW)
123456789N (Start State)	O2XOX6O89C	OXOXXOXOXD
O234X6789N	OXOXXOX8OC	OXOXXOXOXD
OX34X67O9N	OX3OXXO89C	OXOXXOXOXD
OXX4X6OO9N	OXOXXOXO9C	OOXXO0XXD
OX3XXO7O9N	O2XOXXO89C	OXXXO0OXXD
OXXXO0O9N	OXXXO6OOOC	OXOXXOXOXD
OXOXXOXO9N	OOOXX6X89C	OXOXXOXOXD

OX0XX070XN	000XX67X9C	00XXX00XXD
OX30XX709N	000XX678XC	OX00XXXOXD
OX00XXX09N	0XX0XX009C	
OX04X6X09N	02X0XX089C	
OX04X670XN	0004XXX89C	
OX00XX70XN	0004XX7X9C	
OX00X6X0XN	00X0XX0X9C	
02X4X6089N	0004XX78XC	
02XXX0089N	000XX0XX9C	
00XXX00X9N	000XX0X8XC	
00XXX008XN	0004X6XX9C	
023XX0789N	0004X6X8XC	
020XX0X89N	02X0X60X9C	
003XX07X9N	0X0XX0X80C	
003XX078XN	000XX07XXC	
0230XX789N	0230XX0X9C	
0200XXX89N	0000XXX9C	
0204X6X89N	0004X6XX9C	
OX00X6X0XN	0004X67XXC	
0034X67X9N	02X0X60X9C	
00X4X60X9N	0X0XX0X80C	
0204X678XN	000XX07XXC	
	0230XX0X9C	
	0000XXX9C	
	0004X6XX9C	
	0004X67XXC	

2. Transition table DFA untuk komputer yang gerak duluan

COMP GOES FIRST (O)									
NORMAL STATES	INPUTS								
	1	2	3	4	5	6	7	8	9
1234O6789N	XO34O6789N	OX34O6789N	12X4O678ON	O23XO6789N	EEEEEEEEEE	1234OX78ON	O234O6X89N	1234O6OX9N	12O4O678XN
XO34O6789N	EEEEEEEEEE	EEEEEEEEEE	XOX4O67O9C	XO3XO67O9C	EEEEEEEEEE	XO34OX7O9C	XO34O6XO9C	XO34O6OX9N	XO34O67OXC
XO34O6OX9N	EEEEEEEEEE	EEEEEEEEEE	XOX4OOOX9N	XOOXO6OX9C	EEEEEEEEEE	XOO4OXOX9C	EEEEEEEEEE	EEEEEEEEEE	XOO4O6OXXC
XOX4OOOX9N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	XOXXOOOXOD	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	XOXOOOXXC
OX34O6789N	EEEEEEEEEE	EEEEEEEEEE	OXX4O678OC	OX3XO678OC	EEEEEEEEEE	OX34OX78OC	OX34O6X8OC	OX34O67XOC	OXO4O678XN
OXO4O678XN	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXO4O6O8XC	EEEEEEEEEE	OXO4OXO8XC	OXO4O6X8XN	OXO4O6OXXC	EEEEEEEEEE
OXOOO6X8XN	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOOOXOXOD	EEEEEEEEEE	OXOOOXXXD	EEEEEEEEEE
OX34O6XO9N	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OX3XOO789D	EEEEEEEEEE	OX3OOX789D	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE
12X4O678ON	XOX4O678ON	OXX4O678OC	EEEEEEEEEE	O2XO678OC	EEEEEEEEEE	O2X4OX78OC	O2X4O6X8OC	O2X4O67XOC	EEEEEEEEEE
XOX4O678ON	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	XOXO67OOC	EEEEEEEEEE	XOX4OX7OOC	XOX4O6XOOC	XOXOO67XON	EEEEEEEEEE
XOXO67XON	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	XOXOOXOXOD	XOXOOOXOC	EEEEEEEEEE	EEEEEEEEEE
O23XO6789N	EEEEEEEEEE	OX3XO678OC	O2XO678OC	EEEEEEEEEE	EEEEEEEEEE	O23XOX78OC	O23XO6X8OC	O23XO67XOC	O2OXO678XN
O2OXO678XN	EEEEEEEEEE	OXOXO6O8XC	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OOOXOX78XC	OOOXO6X8XC	OOOXO67XXC	EEEEEEEEEE
1234OX78ON	X2O4OX78ON	OX34OX78OC	O2X4OX78OC	O23XOX78OC	EEEEEEEEEE	EEEEEEEEEE	O234OX78OC	O234OX7XOC	EEEEEEEEEE
X2O4OX78ON	EEEEEEEEEE	XXO4OXO8OC	EEEEEEEEEE	X2OXOXO8OC	EEEEEEEEEE	EEEEEEEEEE	X2OOXX8ON	X2O4OXOXC	EEEEEEEEEE
X2OOXX8ON	EEEEEEEEEE	XXOOOXXOOD	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	XXOOOXXXD	EEEEEEEEEE
O234O6X89N	EEEEEEEEEE	OX34O6X8OC	O2X4O6X8OC	O23XO6X8OC	EEEEEEEEEE	O234OX78OC	EEEEEEEEEE	O234O6XOC	O234O6XOXN
O234O6XOXN	EEEEEEEEEE	OX34O6XOXN	OOX4O6XOC	O03XO6XOC	EEEEEEEEEE	OO34OX78OC	EEEEEEEEEE	OO34OX7XOC	O2X4OOOXN
OX34OOXOXN	EEEEEEEEEE	EEEEEEEEEE	OXXOOOXOC	OXO4O6XOC	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE
1234O6OX9N	X2O4O6OX9C	1XO4O6OX9C	O2X4O6OX9N	12OXO6OX9C	EEEEEEEEEE	12O4OXOX9C	EEEEEEEEEE	EEEEEEEEEE	12O4O6OXXC
O2X4O6OX9N	EEEEEEEEEE	OXX4O6OXOC	EEEEEEEEEE	O2XO6OXOC	EEEEEEEEEE	O2X4OXOXOC	EEEEEEEEEE	EEEEEEEEEE	O2X4OOOXN
O2X4OOOXN	EEEEEEEEEE	OXXOOOXXC	EEEEEEEEEE	OOXXOOOXD	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE
12O4O678XN	X2O4O6O8XC	1XO4O6O8XC	EEEEEEEEEE	12OXO6O8XC	EEEEEEEEEE	12O4OXO8XC	12O4O6OXN	12O4O6OXXC	EEEEEEEEEE
12O4O6XOXN	XOO4O6XOC	1XOOO6XOXN	EEEEEEEEEE	1OOXO6XOC	EEEEEEEEEE	1OO4OXOXOC	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE
1XOOO6XOXN	XXOOOXXOC	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE	OXOOOXOXD	EEEEEEEEEE	EEEEEEEEEE	EEEEEEEEEE

List of States :

NORMAL STATES	FINAL STATES (COMPUTER WINS)	FINAL STATES (DRAW)
1234O6789N (START STATE)	X2O4O6OX9C	XXOOOXXOOD
XO34O6789N	X2O4O6O8XC	XOXXOOOXOD
XO34O6OX9N	XOO4O6XOC	OX3XOO789D
XOX4OOOX9N	XXOOOXXOC	OXOXOOXOD
OX34O6789N	OXX4O678OC	OOXXOOOXXD
OXO4O678XN	OX3XO678OC	OXOOOXXOD
OXOOO6X8XN	OXOXO6O8XC	OX3OOX789D
OX34O6XO9N	OX34OX78OC	XOXOOXOXOD
12X4O678ON	XXO4OXO8OC	OXOOOXXOD
XOX4O678ON	OX34O6X8OC	OXOOOXXXD
XOXOO67XON	1XO4O6OX9C	XOOOXXOD
O23XO6789N	OXX4O6OXOC	
O2OXO678XN	OXXOOOXXC	
1234OX78ON	1XO4O6O8XC	
X2O4OX78ON	XOX4O67O9C	
X2OOXX8ON	OXX4O678OC	
O234O6X89N	O2XXO678OC	
O234O6XOXN	O2X4OX78OC	
OX34OOXOXN	O2X4O6X8OC	
1234O6OX9N	OOX4O6XOC	
O2X4O6OX9N	OXXOOOXXC	

02X4000XXN	X03X06709C	
12040678XN	X00X060X9C	
120406X0XN	0X3X06780C	
1X0006X0XN	0X0X0608XC	
	02XX06780C	
	X0XX06700C	
	023X0X780C	
	X20X0X080C	
	023X06X80C	
	003X06X0XC	
	120X060X9C	
	02XX060X0C	
	120X0608XC	
	100X06X0XC	
	X0340X709C	
	X0040X0X9C	
	0X340X780C	
	0X040X08XC	
	02X40X780C	
	X0X40X700C	
	023X0X780C	
	000X0X78XC	
	02340XX80C	
	00340XX0XC	
	12040X0X9C	
	02X40X0X0C	
	12040X08XC	
	10040XX0XC	
	X03406X09C	
	0X3406X80C	
	02X406X80C	
	X0X406X00C	
	X0X000XX0C	
	023X06X80C	
	000X06X8XC	
	02340XX80C	
	0X34067X0C	
	0X04060XXC	
	02X4067X0C	
	023X067X0C	
	000X067XXC	

	O234OX7XOC	
	X2O4OXOXOC	
	O234O6XXOC	
	12O4O6OXXC	
	XO34O67OXC	
	XOO4O6OXXC	
	XOXOOOOXXC	
	12O4O6OXXC	

III. DESKRIPSI DAN KETERANGAN

Alphabets input yang diterima adalah : ['1','2','3','4','5','6','7','8','9']

Representasi states adalah berupa string yang memiliki nilai default berupa :

123456789

Hal ini merepresentasikan keadaan board /*board state* yang dapat diabstraksi menjadi seperti :

1 2 3

4 5 6

7 8 9

Angka diatas dapat dianggap sebagai *index-index* dari lokasi *board* untuk nanti diletakkan simbol-simbol tergantung pemain. Pemain Human memiliki simbol 'X' dan komputer memiliki simbol 'O'. Jadi, andaikan anda akan melakukan gerak dengan memilih tempat ber-index 5, maka akan terjadi :

1 2 3

4 X 6

7 8 9

Keterangan transition table :

- Cells tabel ms. excel yang berwarna hitam dan bertulisan putih mewakili *Start State* dari DFA. Cell ini terletak pada ujung kiri atas pada transition table DFA.
- Cells tabel ms. excel yang berwarna merah dan memiliki karakter bernilai 'EEEEEEEEEE' mewakili states yang *impossible* atau *error* yang dapat berupa seperti : melanggar peraturan bahwa gerakan pertama harus di posisi ke-5 atau memilih petak yang sudah terisi.

- c. Cells tabel ms. excel yang berwarna hijau dan memiliki karakter belakang 'C' mewakili final states yang menunjukkan bahwa komputer telah memenangkan permainan *tic tac toe*.
- d. Cells tabel ms. excel yang berwarna kuning dan memiliki karakter belakang 'D' mewakili final states yang menunjukkan bahwa permainan berakhir pada kondisi *draw*.
- e. Cells table ms. excel yang tidak berwarna dan memiliki karakter belakang 'N' mewakili states normal yang masih dapat menerima input *alphabets* untuk pergi ke state selanjutnya. Ini menunjukkan bahwa permainan belum selesai.

Cara kerja program berkaitan dengan DFA :

1. Program pada awal ketika dijalankan akan menampilkan interface *main menu* serta menampilkan gambaran *boardstate* yang kosong. Lalu program akan meminta input pada User yang berupa 'H' atau 'C' untuk menentukan siapa yang akan melakukan gerakan pertama. 'H' berarti pemain akan melakukan gerak pertama, dan 'C' berarti komputer yang akan melakukan gerak pertama.
2. Ketika 'H' dipilih maka program akan mengarahkan pembacaan file pada file .txt DFA yang bernama 'HFIRST.txt' sedangkan ketika 'C' dipilih maka program akan mengarahkan pembacaan file pada file .txt DFA yang bernama 'CFIRST.txt'.
3. Program akan menggunakan ADT Mesin Kata yang telah dimodifikasi untuk dapat membaca file multiline dan akan membaca file .txt DFA yang telah dibuat. Hasil pembacaan file DFA akan dimasukkan ke matrix yang memiliki elemen bertipe string (array of char di Bahasa C). Matrix ini akan disimpan sebagai transition table DFA.
4. Setelah DFA telah berhasil di-load, maka program siap menjalankan permainan *tic-tac-toe*. Program akan selanjutnya bergerak dengan lambing 'O' (jika komputer giliran pertama) atau akan menunggu inputan dari keyboard user untuk memilih *spot* dimana user akan menaruhkan lambangnya yaitu 'X'. Pengecekan akan dilakukan pada kasus input ini, jika user memilih *spot* di *board* yang melanggar peraturan atau merupakan *spot* yang telah ada isinya, maka akan ditampilkan *error message* dan User akan diminta untuk melakukan input ulang.
5. Setelah didapatkan input yang sesuai dengan ketentuan dan aturan permainan, program akan menyimpan data inputan itu sebagai *alphabet* input bagi DFA. Program selanjutnya akan melakukan konsultasi pada DFA untuk menentukan gerakan terbaik yang harus diambil. Program akan melakukan search di matrix DFA dimana Ia akan mencari data *Board* yang sesuai dengan keadaan *Board* sekarang dan lalu akan mencari *nextState* berdasarkan kolom input dari User. Jika *State* yang didapat memiliki char terakhir yang berupa 'C' maka program akan mengidentifikasi state tersebut sebagai final state dimana komputer menang. Jika *State* yang didapat memiliki char terakhir yang berupa 'D' maka program akan mengidentifikasi state tersebut sebagai final state dimana tidak ada pemenang atau *draw*. Jika *State* yang didapat memiliki char terakhir yang berupa 'N' maka program akan mengidentifikasi state tersebut sebagai state normal yang menunjukkan bahwa permainan masih berlanjut dan akan meminta input user kembali.
6. Setelah didapatkan state baru, state board sekarang akan digantikan dengan state baru tersebut, lalu program akan menentukan kebijakan selanjutnya berdasarkan hasil identifikasi dari karakter terakhir dari state baru tersebut. Jika didapat final state, maka program akan menampilkan tampilan *endgame* yang sesuai dengan keadaan berakhirnya permainan. Jika

didapat bukan final state, maka program akan melanjutkan permainan sampai didapatkan final state.

7. Terakhir, program akan menampilkan semua states yang dilewati oleh program selama berjalanya permainan *tic-tac-toe*.

IV. SOURCE CODE

```
/*PRE-PROCESS SPECIFICATIONS */
#include "mesinkata.h"
#include <stdio.h>
#include <assert.h>

/*-----*/
/*TYPE DECLARATION */
typedef Kata MatriksKata [100][10];
typedef char boardstate[10];
typedef struct {
    boardstate state;
} StateArray[10];

/*-----*/
/*FUNCTIONS & PROCEDURES DECLARATION */
void printBoard(boardstate board);
void printBoard(boardstate board){
    printf("----- CURRENT BOARD STATE -----\\n");
    printf(" |          |\\n");
    for(int i=0; i<=8 ; i++){
        printf("  %c  ", board[i]);
        if (i == 2 || i== 5)
            printf(" |\\n|          |\\n");
    }
    printf(" |\\n|          |");
    printf("\\n");
    printf("----- CURRENT BOARD STATE -----\\n");
    printf("\\n");
}

/*
void ChangeBoardH (boardstate* board, int input);
void ChangeBoardH (boardstate* board, int input){
    *board[input] = 'X';
}

void ChangeBoardC (boardstate* board, MatriksKata DFA, int input, int inputc);
void ChangeBoardC (boardstate* board, MatriksKata DFA, int input, int inputc){
    /* printf("CHANGE COMP BOARD -----\\n");
    printf("i : %d, j : %d\\n", inputc, input);
    for (int k =0; k<=8 ; k++){
        /*printf("%c", DFA[inputc][input].TabKata[k]);
        *board[k] = DFA[inputc][input].TabKata[k];
    }

    printf("NEW BOARD \\n");
    printBoard(*board);
    printf("\\n");
}*/

int SearchState (boardstate board, MatriksKata DFA);
int SearchState (boardstate board, MatriksKata DFA){
    /* DEBUGGER : printf("SEARCH STATE --- \\n");*/
    int i,j,k;
    boolean found = false;
    boolean match;
    j=0;

    for (i=0; i<=32 && (!found) ; i++){
```

```

/* DEBUGGER : printf("PASS ROW : %d",i); */

match = true;
for(k=0; k<=8 && match ; k++){
    /* DEBUGGER : printf("board = %c DFA = %c\n", board[k], DFA[i][j].TabKata[k]); */
    match = board[k] == DFA[i][j].TabKata[k];
}
found = match;
}
/* DEBUGGER : printf("index gotten from search = %d\n", i-1);*/
if (found)
    return i-1;
else
    return -999;
}

void CheckGameState( MatriksKata DFA, int input, int inputc, boolean* Endgame, char *Endstate);
void CheckGameState( MatriksKata DFA, int input, int inputc, boolean* Endgame, char *Endstate){
    /* DEBUGGER : printf("CHECK GAME STATE %c\n", DFA[inputc][input].TabKata[9]);*/

    if ( input <1 || input >9){
        printf("<SYSTEM MESSAGE> Board placement location doesn't exist or out of range!\n");
        *Endstate = 'E';
    }
    else {
        char cond = DFA[inputc][input].TabKata[9];
        *Endstate = cond;
        /* DEBUGGER : printf("ENDSTATE IS : %c\n", *Endstate);*/
        if (cond == 'C' )
            *Endgame = true;
        else if (cond == 'D')
            *Endgame = true;}

}

int DFADecree (boardstate board, MatriksKata DFA, int move, boolean* Endgame, char* Endstate );
int DFADecree (boardstate board, MatriksKata DFA, int move, boolean* Endgame, char* Endstate){
    int idx = SearchState(board, DFA);
    if (idx == -999)
        printf("<SYSTEM MESSAGE> INPUT ERROR!!! \n");
    else
        /* UNUSED : ChangeBoardC(board, DFA, move, idx);*/
        CheckGameState(DFA, move, idx, Endgame, Endstate);
    return idx;
}

/*
void CopyTab (boardstate Tin, boardstate* Tout){
    for (int i=0; i<= 8; i++){
        *Tout[i] = Tin[i];
        printf("Tin :%c Tout :%c", Tin[i], *Tout[i] );
    }
    /*WHY WON'T IT WORK LIKE THIS?
}*/

/*-----*/
/*MAIN PROGRAM*/
int main(){

```

[illegible]

```

boardstate board = {'1','2','3','4','5','6','7','8','9'};
printBoard(board);

/*
board[4] = 'X';
printBoard(board);
printf("\n");

boardstate dummy2;
for (int i=0; i<= 8; i++){
    dummy2[i] = board[i];
    printf("Tin :%c Tout :%c", board[i], dummy2[i] );
}
printBoard(dummy2);
printf("\n");*/

printf("<SYSTEM MESSAGE> : WHO SHALL GO FIRST?\n");
printf("input options : H or C (H means U first, C means computer first)\n");
char firstTurn;
scanf(" %c", &firstTurn);
MatriksKata DFA;
StateArray statelist;
int neff= 0;

if (firstTurn == 'H' || firstTurn == 'h' ){
    char* filename = "HFIRST.txt";
    STARTKATA(filename);
    /*read file into matrix*/
    for (int i =0 ; i<50 && (!EndKata) ; i++){
        for (int j =0; j<=9 ; j++){
            DFA[i][j] = CKata;
            /* DEBUGGER : for (int k =1; k<=10 ; k++){
                printf("%c", CKata.TabKata[k]);
            }
            printf("\n");*/
            ADVKATA();
        }
    }
}

printf("<SYSTEM MESSAGE> INITIATING HUMAN-GOES-FIRST MODULE. SIT BACK TIGHT!\n");
int move,idx;
boolean Endgame = false;
char Endstate;
boardstate dummy;

while(!Endgame) {
    /*printBoard(board);*/
    printf("\n");
    do {
        /*Copying current board state to dummy*/
        for (int i=0; i<= 8; i++){
            dummy[i] = board[i];
        }

        /* Copying dummy board to list of state */
        for (int i=0; i<= 8; i++){
            statelist[neff].state[i] = dummy[i];
        }

        /*Asking for user move */
        printf("<SYSTEM MESSAGE> Select a spot on the board...\n");
        scanf("%d", &move);

```

```

while(getchar() != '\n'); /*SCANF IS SO FUCKING FLAWED, NEED THIS TO DISCARD BUFFER CLUTTERS*/

/*Consultates to DFA */

printf("<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...\\n");
idx = DFADecree(dummy, DFA, move, &Endgame, &Endstate);
/*DEBUGGER : printf("%d\\n",idx); */
if (idx == -999 || Endstate == 'E') {
    /* DEBUGGER : printf("idx = %d, endstate = %c", idx, Endstate);*/
    printf("<SYSTEM MESSAGE> INPUT ERROR, REINPUT! \\n");
}
else {
    /*copying boardstate from DFA to dummy */
    for (int k = 0; k <= 8; k++){
        /*DEBUGGER : printf("%c", DFA[idx][move].TabKata[k]);*/
        dummy[k] = DFA[idx][move].TabKata[k];
    }
    /*dummy[move-1] = 'X';*/
}

} while (Endstate == 'E' || idx == -999); /*Asking user for input until dummy board state is acceptable */

/*Status : User input already accepted. */
neff++; /* neff of states-passed list is incremented */

/*Dummy is in acceptable condition. Dummy is copied to real board */
for (int i = 0; i <= 8; i++){
    board[i] = dummy[i];
}

/*printing the board */
printf("<SYSTEM MESSAGE> RESULT : \\n");
printBoard(board);
printf("\\n");

/*Final state check */
if (Endgame){
    if (Endstate == 'C'){
        printf(" _____\\n");
        printf("| . _____) |\\n");
        printf("| | / / | |\\n");
        printf("| | / / | |\\n");
        printf("| | / | | .-\\n");
        printf("| | / | / _ \\ \\n");
        printf("| | | | ' , |\\n");
        printf("| | (\\ \\ _ .\\n");
        printf("| | .- ' - .\\n");
        printf("| | /Y . . Y \\n");
        printf("| | // | | \\ \\n");
        printf("| | // | . | \\ \\n");
        printf("| | ' ) | | ('\\n");
        printf("| | | | ' |\\n");
        printf("| | | | |\\n");
        printf("| | | | |\\n");
        printf("| | | | |\\n");
        printf("| | / | |\\n");
        printf("~~~~~|_ ' ' _|~~~~~\\n");
        printf("|~|~~~~~\\ \\ ~~~~~\\n");
        printf("| | \\ \\ |\\n");
        printf("| : \\ \\ : :\\n");
        printf("| . ' . \\n");
    }
}

```

```

        printf("<VERDICT> YOU HAVE BEEN EXECUTED (FIGURATIVELY) BY COMPUTER!(YOU LOSE) \n");
    }
    else if (Endstate == 'D'){
        printf("~~~~~Congratulations~~~~~\n");
        printf("<VERDICT> DRAWWWWWWWW!!! \n");
        printf("YOU THOUGHT U COULD WIN? \n");
        printf("KEEP DREAMING! \n");
    }
}
}
}

/*COMP GOES FIRST MODULE */
else if (firstTurn == 'C' || firstTurn == 'c'){
    /*DFA file is read and stored in matrikskata */
    char* filename = "CFIRST.txt";
    STARTKATA(filename);
    for (int i = 0 ; i < 100 && (!EndKata) ; i++){
        for (int j = 0; j <= 9 ; j++){
            DFA[i][j] = CKata;
            /*DEBUGGER for (int k = 1; k <= 10 ; k++){
                printf("%c", CKata.TabKata[k]);
                printf("\n");*/
            ADVKATA();
        }
    }

    /*variable initializations */
    printf("<SYSTEM MESSAGE> INITIALIZING COMP-GOES-FIRST MODULE \n");
    int move,idx;
    boolean Endgame = false;
    char Endstate;
    boardstate dummy;

    /*Computer takes the first move */
    printf("<SYSTEM MESSAGE> COMPUTER MAKES THE FIRST MOVE...\n");
    board[4] = 'O';
    printBoard(board);
    printf("\n");

    /*Copying current board to states-passed list */
    for (int i=0; i<= 8; i++){
        statelist[neff].state[i] = dummy[i];
    }

    /*INPUT LOOP*/
    while(!Endgame) {
        do {
            /*copy current board state to dummy */
            for (int i=0; i<= 8; i++){
                dummy[i] = board[i];
            }

            /*copying current board state to states-passed list */
            for (int i=0; i<= 8; i++){
                statelist[neff].state[i] = dummy[i];
            }

            /*asking for user input */
            printf("<SYSTEM MESSAGE> Select a spot on the board...\n");
            scanf("%d", &move);
            while(getchar() != '\n'); /*SCANF IS SO FUCKING FLAWED, NEED THIS TO DISCARD BUFFER CLUTTERS*/

```

```

/*Computer Consultates to DFA*/
printf("<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...\n");
idx = DFADecree(dummy, DFA, move, &Endgame, &Endstate);

if (idx == -999 || Endstate == 'E') {
    printf("<SYSTEM MESSAGE> INPUT ERROR, REINPUT! \n");
}
else {
    /*copying boardstate from DFA to dummy */
    for (int k = 0; k <= 8 ; k++){
        /*DEBUGGER : printf("%c", DFA[idx][move].TabKata[k]);*/
        dummy[k] = DFA[idx][move].TabKata[k];
    }
}

} while (Endstate == 'E' || idx == -999);

/*Status : User input already accepted. */
neff++; /* neff of states-passed list is incremented */

/*Dummy is in acceptable condition. Dummy is copied to real board */
for (int i = 0; i <= 8; i++){
    board[i] = dummy[i];
}

/*printing the board */
printf("<SYSTEM MESSAGE> RESULT : \n");
printBoard(board);
printf("\n");

/*Final state check */
if (Endgame){
    if (Endstate == 'C'){
        printf(" _____\n");
        printf(" | . _____ ) | \n");
        printf(" | | / / | | \n");
        printf(" | | / | | \n");
        printf(" | | / | | . \n");
        printf(" | | / | | _ \n");
        printf(" | | | | ' , | \n");
        printf(" | | ( \ \ _ \n");
        printf(" | | . - ' \n");
        printf(" | | / Y . . Y \n");
        printf(" | | // | | \ \ \n");
        printf(" | | // | . | \ \ \n");
        printf(" | | ' ) | | ( \n");
        printf(" | | | | ' | \n");
        printf(" | | | | \n");
        printf(" | | | | \n");
        printf(" | | | | \n");
        printf(" | | / | | \ \ \n");
        printf(" | ~ | ~ ~ ~ ~ ~ | _ ' - ' | ~ ~ ~ ~ ~ \n");
        printf(" | ~ | ~ ~ ~ ~ ~ \ \ ~ ~ \n");
        printf(" | | \ \ | \n");
        printf(" | : \ \ : : \n");
        printf(" | . ' . \n");
        printf("<VERDICT> YOU HAVE BEEN EXECUTED (FIGURATIVELY) BY COMPUTER!(YOU LOSE) \n");
    }
    else if (Endstate == 'D'){

```



```

        printf("~~~Congratulations~~~\n");
        printf("<VERDICT> DRAWWWWWWW!!! \n");
        printf("YOU THOUGHT U COULD WIN? \n");
        printf("KEEP DREAMING!\n\n");
    }
}
}
}
else{ /*TURN INPUT IS WRONG */
    printf("<SYSTEM ERROR MESSAGE> INPUT IS NOT VALID! PLEASE READ CAREFULLY!!! _-\n");}

/*Copying current board state to states-passed list */
for (int i=0; i<= 8; i++){
    statelist[neff].state[i] = board[i];
}

/*IF Turn Input is wrong, offensive comments ensue ...*/
if(neff ==0 ) {
    printf("Seriously?\n");
    printf("    ,\n");
    printf("( ^-./ ( .....,\n");
    printf(" ^-__ / ``:.. /7_-\n");
    printf(" ^-.- - - `::' .- ( ^-=\n");
    printf(" \\ ^-__ |/_ ?' ^- ^-\n");
    printf(" \\ - /)----- ^-\n");
    printf(" ^-.. ^-,' ^-\n");
    printf("    \ ^-')---\n");
    printf("    ) )\n");
    printf("    | _|\n");
    printf("    ( \ \n");
    printf("    L /\n");
    printf("    | \ \n");
    printf("    )_ _ \ \n");
    printf("    \ \ ^- ^-\n");
    printf("    L | \n");
    printf("    | \ \ | \n");
    printf("    \ \ L ) \ \n");
    printf("    L_ ( / \n");
    printf("    | \ \ . \ \n");
    printf("    ... | ^ \ \n");
    printf("    _- ^-== ^ \ \n");
    printf("    ( _ ^ ^-\ \ \n");
    printf("    ^- ^ ^ | \n");
    printf("    ( _ /\n");
    printf("JANGAN SENGAJA BIKIN ERROR...!!!\n ");}

else {
    /*Print States-passed list */
    printf("~~~~~\n");
    printf("STATES PASSED : \n");
    for (int i=0; i<= neff; i++){
        printf("[%d]. ",i+1);
        for (int j=0; j<= 8; j++){
            printf("%c",statelist[i].state[j]);
        }
        printf("\n");
    }
}
}
/*goto BEGIN;*/
}

```

V. CONTOH MASUKAN DAN KELUARAN

0. Interface Awal

[illegible]

- Inputs : H, 5, 3, 2

```

----- CURRENT BOARD STATE -----
| 1           2           3           |
| 4           5           6           |
| 7           8           9           |
|-----|
<SYSTEM MESSAGE> : WHO SHALL GO FIRST?
input options : H or C (H means U first, C means computer first)
H
<SYSTEM MESSAGE> INITIATING HUMAN-GOES-FIRST MODULE. SIT BACK TIGHT!
<SYSTEM MESSAGE> Select a spot on the board...
5
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
| 0           2           3           |
| 4           X           6           |
| 7           8           9           |
|-----|
----- CURRENT BOARD STATE -----

<SYSTEM MESSAGE> Select a spot on the board...
3
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
| 0           2           X           |
| 4           X           6           |
| 0           8           9           |
|-----|
----- CURRENT BOARD STATE -----

```

[illegible]

2. Pemain gerak duluan dan draw

Inputs : H, 3, 4, 8, 9

```

<SYSTEM MESSAGE> : WHO SHALL GO FIRST?
input options : H or C (H means U first, C means computer first)
H
<SYSTEM MESSAGE> INITIATING HUMAN-GOES-FIRST MODULE. SIT BACK TIGHT!

<SYSTEM MESSAGE> Select a spot on the board...
5
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
|
| 0      2      3      |
| 4      X      6      |
| 7      8      9      |
|
|----- CURRENT BOARD STATE -----

<SYSTEM MESSAGE> Select a spot on the board...
3
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
|
| 0      2      X      |
| 4      X      6      |
| 0      8      9      |
|
|----- CURRENT BOARD STATE -----

<SYSTEM MESSAGE> Select a spot on the board...
4
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
|
| 0      2      X      |
| X      X      0      |
| 0      8      9      |
|
|----- CURRENT BOARD STATE -----

<SYSTEM MESSAGE> Select a spot on the board...
8
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
|
| 0      0      X      |
| X      X      0      |
| 0      X      9      |
|
|----- CURRENT BOARD STATE -----

<SYSTEM MESSAGE> Select a spot on the board...
9
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
|
| 0      0      X      |
| X      X      0      |
| 0      X      X      |
|
|----- CURRENT BOARD STATE -----

~~~~~Congratulations~~~~~
<VERDICT> DRAWMMMMMM!!!
YOU THOUGHT U COULD WIN?
KEEP DREAMING!
~~~~~
STATES PASSED :
[1]. 123456789
[2]. 0234X6789
[3]. 02X4X6089
[4]. 02XXX0089
[5]. 00XXX00X9
[6]. 00XXX00XX

Process returned 0 (0x0)   execution time : 18.511 s
Press any key to continue.

```

- Inputs : C, 1, 3

```

<SYSTEM MESSAGE>: WHO SHALL GO FIRST?
input options : H or C (H means U first, C means computer first)
C
<SYSTEM MESSAGE> INITIALIZING COMP-GOES-FIRST MODULE
<SYSTEM MESSAGE> COMPUTER MAKES THE FIRST MOVE...
----- CURRENT BOARD STATE -----
|
| 1          2          3          |
|
| 4          0          6          |
|
| 7          8          9          |
|
|----- CURRENT BOARD STATE -----|
|
<SYSTEM MESSAGE> Select a spot on the board...
1
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
|
| X          0          3          |
|
| 4          0          6          |
|
| 7          8          9          |
|
|----- CURRENT BOARD STATE -----|
|
<SYSTEM MESSAGE> Select a spot on the board...
3
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
|
| X          0          X          |
|
| 4          0          6          |
|
| 7          0          9          |
|
|----- CURRENT BOARD STATE -----|

```

[illegible]

4. Komputer duluan dan draw

Inputs : C, 3, 1, 8, 6

```

<SYSTEM MESSAGE> : WHO SHALL GO FIRST?
input options : H or C <H means U first, C means computer first>
C
<SYSTEM MESSAGE> INITIALIZING COMP-GOES-FIRST MODULE
<SYSTEM MESSAGE> COMPUTER MAKES THE FIRST MOVE...
----- CURRENT BOARD STATE -----
| 1      2      3 |
| 4      0      6 |
| 7      8      9 |
|_|_|_|
----- CURRENT BOARD STATE -----

<SYSTEM MESSAGE> Select a spot on the board...
3
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
| 1      2      X |
| 4      0      6 |
| 7      8      0 |
|_|_|_|
----- CURRENT BOARD STATE -----

<SYSTEM MESSAGE> Select a spot on the board...
1
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
| X      0      X |
| 4      0      6 |
| 7      8      0 |
|_|_|_|
----- CURRENT BOARD STATE -----

<SYSTEM MESSAGE> Select a spot on the board...
8
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
| X      0      X |
| 0      0      6 |
| 7      X      0 |
|_|_|_|
----- CURRENT BOARD STATE -----

<SYSTEM MESSAGE> Select a spot on the board...
6
<SYSTEM MESSAGE> COMPUTER IS CONSULTING TO DFA ...
<SYSTEM MESSAGE> RESULT :
----- CURRENT BOARD STATE -----
| X      0      X |
| 0      0      X |
| 0      X      0 |
|_|_|_|
----- CURRENT BOARD STATE -----

~~~~~Congratulations~~~~~
<VERDICT> DRAWWWWWW!!!
YOU THOUGHT U COULD WIN?
KEEP DREAMING!

~~~~~
STATES PASSED :
[1]. 123406789
[2]. 12X406780
[3]. X0X406780
[4]. X0X0067X0
[5]. X0X00X0X0

Process returned 0 (0x0)   execution time : 12.270 s
Press any key to continue.

```

