# 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**

1. **DESKRIPSI TUGAS**

|  |  |
| --- | --- |
| ***Tic-tac-toe*** | |
| [Tic tac toe.svg](https://ms.wikipedia.org/wiki/Fail:Tic_tac_toe.svg)  Penyudah kepada Tic-tac-toe | |
| **Genre** | [Permainan atas kertas](https://ms.wikipedia.org/wiki/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 dalampermainan ini.

1. **NOTASI DFA**
2. **Transition Table untuk DFA jika pemain gerak duluan**

****

**List of States :**

|  |  |  |
| --- | --- | --- |
| **NORMAL STATES** | **FINAL STATES (COMPUTER WINS)** | **FINAL STATES (DRAW)** |
| **123456789N (Start State)** | **O2XOX6O89C** | **OXOOXXXOXD** |
| **O234X6789N** | **OXOXXOX8OC** | **OXOXXOXOXD** |
| **OX34X67O9N** | **OX3OXXO89C** | **OXOOXXXOXD** |
| **OXX4X6OO9N** | **OXXOXXOO9C** | **OOXXXOOXXD** |
| **OX3XXO7O9N** | **O2XOXXO89C** | **OXXXXOOOXD** |
| **OXXXXOOO9N** | **OXXXX6OOOC** | **OXOXXOXOXD** |
| **OXOXXOXO9N** | **OOOXX6X89C** | **OXOOXXXOXD** |
| **OXOXXO7OXN** | **OOOXX67X9C** | **OOXXXOOXXD** |
| **OX3OXX7O9N** | **OOOXX678XC** | **OXOOXXXOXD** |
| **OXOOXXXO9N** | **OXXOXXOO9C** |  |
| **OXO4X6XO9N** | **O2XOXXO89C** |  |
| **OXO4X67OXN** | **OOO4XXX89C** |  |
| **OXOOXX7OXN** | **OOO4XX7X9C** |  |
| **OXOOX6XOXN** | **OOXOXXOX9C** |  |
| **O2X4X6O89N** | **OOO4XX78XC** |  |
| **O2XXXOO89N** | **OOOXXOXX9C** |  |
| **OOXXXOOX9N** | **OOOXXOX8XC** |  |
| **OOXXXOO8XN** | **OOO4X6XX9C** |  |
| **O23XXO789N** | **OOO4X6X8XC** |  |
| **O2OXXOX89N** | **O2XOX6OX9C** |  |
| **OO3XXO7X9N** | **OXOXXOX8OC** |  |
| **OO3XXO78XN** | **OOOXXO7XXC** |  |
| **O23OXX789N** | **O23OXXOX9C** |  |
| **O2OOXXX89N** | **OOOOXXXX9C** |  |
| **O2O4X6X89N** | **O0O4X6XX9C** |  |
| **OXOOX6XOXN** | **OOO4X67XXC** |  |
| **OO34X67X9N** | **O2XOX6OX9C** |  |
| **OOX4X6OX9N** | **OXOXXOX8OC** |  |
| **O2O4X678XN** | **OOOXXO7XXC** |  |
|  | **O23OXXOX9C** |  |
|  | **OOOOXXXX9C** |  |
|  | **O0O4X6XX9C** |  |
|  | **OOO4X67XXC** |  |

1. **Transition table DFA untuk komputer yang gerak duluan**

****

**List of States :**

|  |  |  |
| --- | --- | --- |
| **NORMAL STATES** | **FINAL STATES (COMPUTER WINS)** | **FINAL STATES (DRAW)** |
| **1234O6789N (START STATE)** | **X2O4O6OX9C** | **XXOOOXXOOD** |
| **XO34O6789N** | **X2O4O6O8XC** | **XOXXOOOXOD** |
| **XO34O6OX9N** | **XOO4O6XOXC** | **OX3XOO789D** |
| **XOX4OOOX9N** | **XXOOOOXOXC** | **OXOXOOXOXD** |
| **OX34O6789N** | **OXX4O678OC** | **OOXXOOOXXD** |
| **OXO4O678XN** | **OX3XO678OC** | **OXOOOXXOXD** |
| **OXOOO6X8XN** | **OXOXO6O8XC** | **OX3OOX789D** |
| **OX34O6XO9N** | **OX34OX78OC** | **XOXOOXOXOD** |
| **12X4O678ON** | **XXO4OXO8OC** | **OXOOOXXOXD** |
| **XOX4O678ON** | **OX34O6X8OC** | **OXOOOOXXXD** |
| **XOXOO67XON** | **1XO4O6OX9C** | **XOOOOXXXOD** |
| **O23XO6789N** | **OXX4O6OXOC** |  |
| **O2OXO678XN** | **OXXOOOOXXC** |  |
| **1234OX78ON** | **1XO4O6O8XC** |  |
| **X2O4OX78ON** | **XOX4O67O9C** |  |
| **X2OOOXX8ON** | **OXX4O678OC** |  |
| **O234O6X89N** | **O2XXO678OC** |  |
| **O234O6XOXN** | **O2X4OX78OC** |  |
| **OX34OOXOXN** | **O2X4O6X8OC** |  |
| **1234O6OX9N** | **OOX4O6XOXC** |  |
| **O2X4O6OX9N** | **OXXOOOXOXC** |  |
| **O2X4OOOXXN** | **XO3XO67O9C** |  |
| **12O4O678XN** | **XOOXO6OX9C** |  |
| **12O4O6XOXN** | **OX3XO678OC** |  |
| **1XOOO6XOXN** | **OXOXO6O8XC** |  |
|  | **O2XXO678OC** |  |
|  | **XOXXO67OOC** |  |
|  | **O23XOX78OC** |  |
|  | **X2OXOXO8OC** |  |
|  | **O23XO6X8OC** |  |
|  | **OO3XO6XOXC** |  |
|  | **12OXO6OX9C** |  |
|  | **O2XXO6OXOC** |  |
|  | **12OXO6O8XC** |  |
|  | **1OOXO6XOXC** |  |
|  | **XO34OX7O9C** |  |
|  | **XOO4OXOX9C** |  |
|  | **OX34OX78OC** |  |
|  | **OXO4OXO8XC** |  |
|  | **O2X4OX78OC** |  |
|  | **XOX4OX7OOC** |  |
|  | **O23XOX78OC** |  |
|  | **OOOXOX78XC** |  |
|  | **O234OXX8OC** |  |
|  | **OO34OXXOXC** |  |
|  | **12O4OXOX9C** |  |
|  | **O2X4OXOXOC** |  |
|  | **12O4OXO8XC** |  |
|  | **1OO4OXXOXC** |  |
|  | **XO34O6XO9C** |  |
|  | **OX34O6X8OC** |  |
|  | **O2X4O6X8OC** |  |
|  | **XOX4O6XOOC** |  |
|  | **XOXOOOXXOC** |  |
|  | **O23XO6X8OC** |  |
|  | **OOOXO6X8XC** |  |
|  | **O234OXX8OC** |  |
|  | **OX34O67XOC** |  |
|  | **OXO4O6OXXC** |  |
|  | **O2X4O67XOC** |  |
|  | **O23XO67XOC** |  |
|  | **OOOXO67XXC** |  |
|  | **O234OX7XOC** |  |
|  | **X2O4OXOXOC** |  |
|  | **O234O6XXOC** |  |
|  | **12O4O6OXXC** |  |
|  | **XO34O67OXC** |  |
|  | **XOO4O6OXXC** |  |
|  | **XOXOOOOXXC** |  |
|  | **12O4O6OXXC** |  |

1. **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 :

1. Cells tabel ms. excel yang berwarna hitam dan bertulisan putih mewakili *Start State* dari DFA. Cell ini berletak pada ujung kiri atas pada transition table DFA.
2. 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.
3. 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*.
4. Cells tabel ms. excel yang berwarna kuning dan memiliki karakter belakang ‘D’ mewakili final states yang menunjukkan bahwa permainan berakhir pada kondisi *draw*.
5. 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 permain 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*.
8. **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(){

/\*BEGIN:\*/

/\*MAIN MENU INTERFACE \*/

printf(" \n");

printf("TTTTTTTTTTTTTTTTTTTTTTTIIIIIIIIII CCCCCCCCCCCCC \n");

printf("T:::::::::::::::::::::TI::::::::I CCC::::::::::::C \n");

printf("T:::::::::::::::::::::TI::::::::I CC:::::::::::::::C \n");

printf("T:::::TT:::::::TT:::::TII::::::II C:::::CCCCCCCC::::C \n");

printf("TTTTTT T:::::T TTTTTT I::::I C:::::C CCCCCC \n");

printf(" T:::::T I::::I C:::::C \n");

printf(" T:::::T I::::I C:::::C \n");

printf(" T:::::T I::::I C:::::C \n");

printf(" T:::::T I::::I C:::::C \n");

printf(" T:::::T I::::I C:::::C \n");

printf(" T:::::T I::::I C:::::C \n");

printf(" T:::::T I::::I C:::::C CCCCCC \n");

printf(" TT:::::::TT II::::::II C:::::CCCCCCCC::::C \n");

printf(" T:::::::::T I::::::::I CC:::::::::::::::C \n");

printf(" T:::::::::T I::::::::I CCC::::::::::::C \n");

printf(" TTTTTTTTTTT IIIIIIIIII CCCCCCCCCCCCC to the \n");

printf(" \n");

printf(" \n");

printf(" tttt \n");

printf(" ttt:::t \n");

printf(" t:::::t \n");

printf(" t:::::t \n");

printf("ttttttt:::::ttttttt aaaaaaaaaaaaa cccccccccccccccc \n");

printf("t:::::::::::::::::t a::::::::::::a cc:::::::::::::::c \n");

printf("t:::::::::::::::::t aaaaaaaaa:::::a c:::::::::::::::::c \n");

printf("tttttt:::::::tttttt a::::a c:::::::cccccc:::::c \n");

printf(" t:::::t aaaaaaa:::::a c::::::c ccccccc \n");

printf(" t:::::t aa::::::::::::a c:::::c \n");

printf(" t:::::t a::::aaaa::::::a c:::::c \n");

printf(" t:::::t tttttta::::a a:::::a c::::::c ccccccc \n");

printf(" t::::::tttt:::::ta::::a a:::::a c:::::::cccccc:::::c \n");

printf(" tt::::::::::::::ta:::::aaaa::::::a c:::::::::::::::::c \n");

printf(" tt:::::::::::tt a::::::::::aa:::a cc:::::::::::::::c \n");

printf(" ttttttttttt aaaaaaaaaa aaaa cccccccccccccccc to the\n");

printf(" \n");

printf(" \n");

printf("TTTTTTTTTTTTTTTTTTTTTTT EEEEEEEEEEEEEEEEEEEEEE \n");

printf("T:::::::::::::::::::::T E::::::::::::::::::::E \n");

printf("T:::::::::::::::::::::T E::::::::::::::::::::E \n");

printf("T:::::TT:::::::TT:::::T EE::::::EEEEEEEEE::::E \n");

printf("TTTTTT T:::::T TTTTTT ooooooooooo E:::::E EEEEEE \n");

printf(" T:::::T oo:::::::::::oo E:::::E \n");

printf(" T:::::T o:::::::::::::::o E::::::EEEEEEEEEE \n");

printf(" T:::::T o:::::ooooo:::::o E:::::::::::::::E \n");

printf(" T:::::T o::::o o::::o E:::::::::::::::E \n");

printf(" T:::::T o::::o o::::o E::::::EEEEEEEEEE \n");

printf(" T:::::T o::::o o::::o E:::::E \n");

printf(" T:::::T o::::o o::::o E:::::E EEEEEE \n");

printf(" TT:::::::TT o:::::ooooo:::::oEE::::::EEEEEEEE:::::E \n");

printf(" T:::::::::T o:::::::::::::::oE::::::::::::::::::::E \n");

printf(" T:::::::::T oo:::::::::::oo E::::::::::::::::::::E \n");

printf(" TTTTTTTTTTT ooooooooooo EEEEEEEEEEEEEEEEEEEEEE \n");

printf(" \n");

printf(" \n");

printf("~~~~~WELCOME TO THE-TIC-TO-THE-TAC-TO-THE-TOE~~~~~~\n");

printf("~A FANTASTIC GAME WHERE YOU CAN\'T WIN AT ALL~\n");

printf("\n");

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> DRAWWWWWWW!!! \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");

}

}

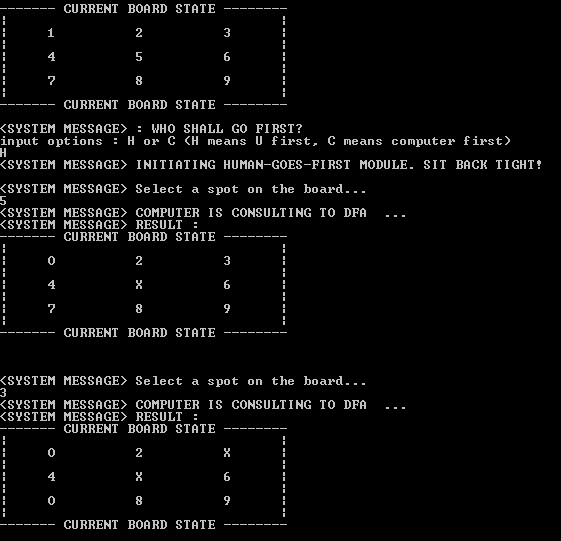
/\*goto BEGIN;\*/

}

1. **CONTOH MASUKAN DAN KELUARAN**
2. Interface Awal



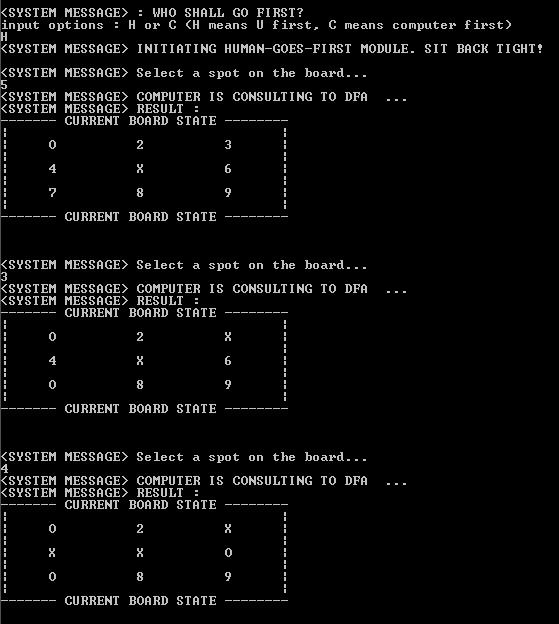
1. Pemain gerak duluan dan kalah

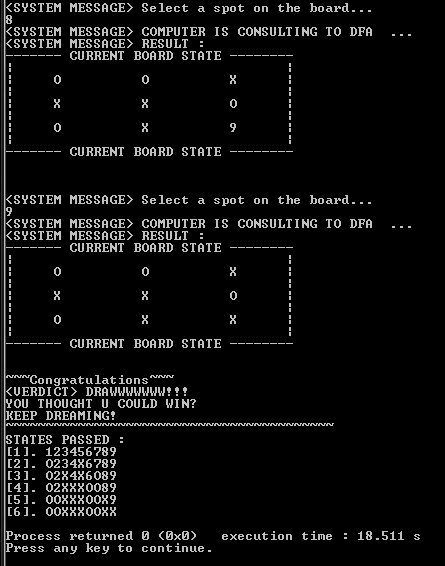
Inputs : H, 5, 3, 2



1. Pemain gerak duluan dan draw

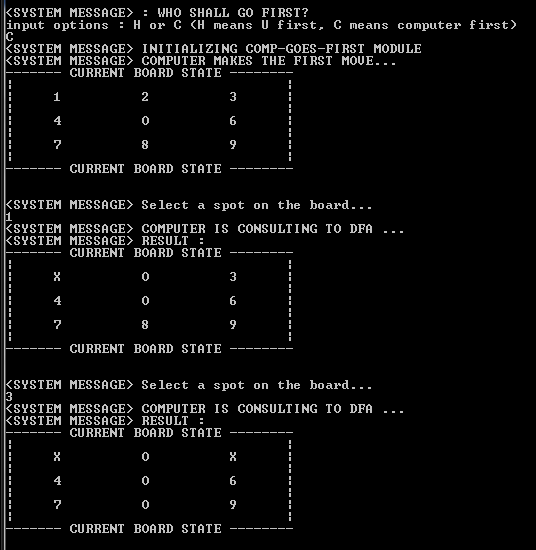
Inputs : H, 3, 4, 8, 9





1. Komputer duluan dan pemain kalah

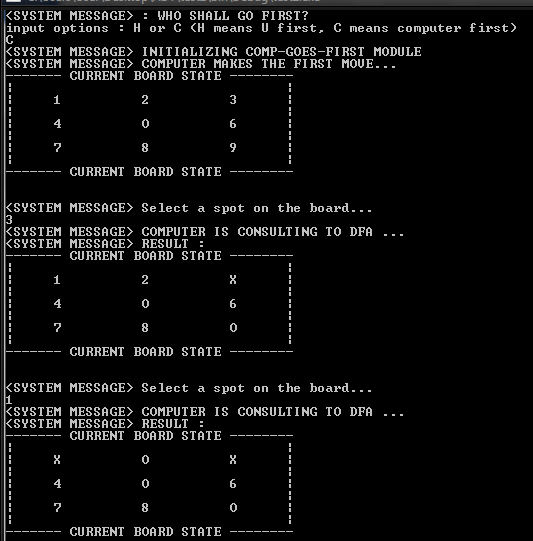
Inputs : C, 1, 3

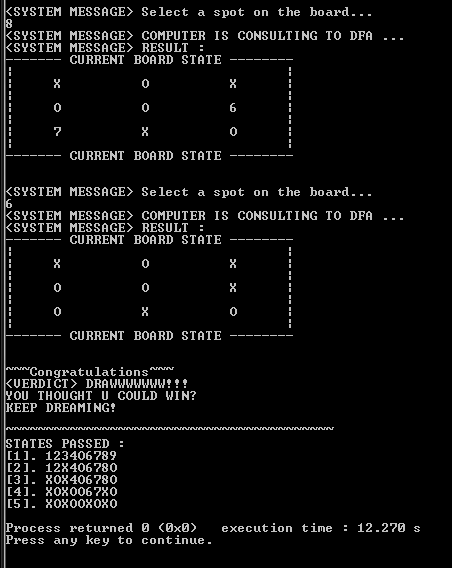




1. Komputer duluan dan draw

Inputs : C, 3, 1, 8, 6





1. Input Error 1
2. Input Error 2

