**TUGAS BESAR 1**

**IF2220 – TEORI BAHASA FORMAL DAN OTOMATA**

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

Disusun oleh

Saskia Imani

13517142

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK INFORMATIKA DAN ELEKTRO

INSTITUT TEKNOLOGI BANDUNG

1. **DESKRIPSI PERMASALAHAN**

*Tic-tac-toe* ​adalah sebuah permainan melibatkan dua orang yang secara bergiliran saling membuat huruf X dan O didalam sebuah kotak berukuran 3 x 3. Pemenang dari permainan ini adalah pemain pertama yang berhasil membuat tiga tanda “X” atau tanda “O” berurutan di dalam suatu permainan.

Pada tugas ini, dibuat sebuah permainan *tic-tac-toe* sederhana yang akan dimainkan oleh komputer dan seorang player. Program harus bisa memastikan 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 diwajibkan untuk membaca konfigurasi dari file eksternal, dan logika state machine tidak boleh di-*hardcode* ke program secara langsung.

1. **DFA**

Pada program ini, digunakan sebuah DFA untuk menentukan transisi yang akan dialami oleh *state* program, dengan rancangan sebagai berikut:



\*

\*

\*

\*

\*



\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*



\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*



\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*



\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

Dalam pembuatannya, tabel transisi ini menggunakan MS Excel, dan sebelum dibentuk menjadi tabel berisi nama *state* saja, dilakukan gambaran setiap kotak dalam permainan terlebih dahulu untuk membayangkan perubahan yang terjadi secara jelas. Gambaran tersebut kemudian dipecah menjadi tabel transisi di atas dan sebuah kamus transisi yang menggambarkan isi setiap kotak dalam permainan ketika *state* adalah *state* tertentu. Berikut adalah kamus *state*:

   

Pada tabel di atas, *state* menang diberi warna hijau sedangkan *state* seri diberi warna biru.

1. ***STATE* DAN AKSI**

*State* pada DFA merupakan *state* di mana pemain baru saja meletakkan X berikutnya dan komputer kemudian membalas gerakan pemain. Sebagai contoh, *state* q001 merupakan *state* di mana komputer mulai terlebih dahulu. Pemain kemudian dapat meletakkan X di posisi 1-4 atau 6-9. Misalkan pemain meletakkan X di 2, maka komputer akan membalas gerakan tersebut dengan meletakkan O di posisi 7. *State* berubah menjadi q003.



Gerakan komputer ditentukan berdasarkan gerakan yang terbaik untuk memenangkan atau membuat seri permainan tic-tac-toe secara cepat. Referensi untuk gerakan komputer terbaik diambil dari Wikipedia dan Quora.

Dalam program digunakan sebuah fungsi yang diberi nama Trans untuk menentukan transisi, dengan parameter CState dan x, di mana CState adalah *state* permainan saat ini dan x adalah indeks transisi yang dicari. Indeks transisi untuk p (kondisi ketika pemain mulai terlebih dahulu) adalah -1 dan c (kondisi ketika komputer mulai terlebih dahulu) adalah 0, sedangkan untuk x = 1...9 indeks adalah x + 2. Untuk fungsi Trans secara lengkap, serta fungsi-fungsi lainnya, terdapat pada bagian *source code* laporan ini.

1. **SOURCE CODE**

Source code program terdiri dari sebuah program utama yang menjalankan fungsi Play, serta fungsi-fungsi sebagai berikut:

* 1. **Prosedur Print**

Prosedur Print adalah fungsi yang digunakan untuk membaca kamus state pada file eksternal dan mencetak gambaran setiap kotak permainan dalam bentuk *grid* pada saat *state* tertentu. Prosedur ini memiliki parameter CState, yaitu *state* permainan pada saat ini.

void Print (char \* state) {

FILE \*fp;

char MARK[6], label[5], content[10];

fp = fopen("file.txt", "r");

fscanf(fp, "%s", MARK);

while (strcmp(MARK, "KAMUS") != 0) {

fscanf(fp, "%s", MARK);

}

fscanf(fp, "%s", label);

while (strcmp(label, state) != 0) {

fscanf(fp, "%s", label);

fscanf(fp, "%s", label);

}

fscanf(fp, "%s", content);

printf("%c %c %c\n", content[0], content[1], content[2]);

printf("%c %c %c\n", content[3], content[4], content[5]);

printf("%c %c %c\n", content[6], content[7], content[8]);

};

* 1. **Fungsi Trans**

Fungsi Trans adalah fungsi yang digunakan untuk membaca tabel transisi pada file eksternal dan menentukan transisi *state* permainan setelah pemain memasukkan gerakannya selanjutnya. Fungsi ini memiliki parameter CState dan x, di mana CState adalah *state* permainan pada saat ini dan x adalah indeks transisi. Indeks transisi untuk p adalah -1 dan c adalah 0, sedangkan untuk x = 1...9 indeks adalah x + 2. Keluaran dari fungsi ini berupa “string” yang berisi *state* permainan yang sudah diubah.

char\* Trans (char \* state, int x) {

FILE \*fp;

char MARK[9], skip[2], s[5], to[5], \*final;

fp = fopen("file.txt", "r");

fscanf(fp, "%s", MARK);

while (strcmp(MARK, "TRANSISI") != 0) {

fscanf(fp, "%s", MARK);

}

for (int i = 1; i <= 11; i++) {

fscanf(fp, "%s", skip);

}

fscanf(fp, "%s", s);

while (strcmp(s, state) != 0) {

fscanf(fp, "%s", s); //transisi p

fscanf(fp, "%s", s); //transisi c

fscanf(fp, "%s", s); //transisi 1

fscanf(fp, "%s", s); //transisi 2

fscanf(fp, "%s", s); //transisi 3

fscanf(fp, "%s", s); //transisi 4

fscanf(fp, "%s", s); //transisi 5

fscanf(fp, "%s", s); //transisi 6

fscanf(fp, "%s", s); //transisi 7

fscanf(fp, "%s", s); //transisi 8

fscanf(fp, "%s", s); //transisi 9

fscanf(fp, "%s", s); //state berikutnya

}

for (int i = 1; i <= x+2; i++) {

fscanf(fp, "%s", to);

}

if (strcmp(to, "----") != 0) {

final = to;

} else {

printf("That spot is occupied.\n");

final = state;

}

return final;

}

* 1. **Prosedur CheckWin**

Prosedur CheckWin adalah fungsi yang digunakan untuk membaca daftar *state* menang pada file eksternal dan menetukan apakah *state* saat ini merupakan *state* di mana komputer sudah menang. Prosedur ini memiliki parameter CState dan win, di mana CState adalah *state* permainan pada saat ini dan win adalah “boolean” yang menyatakan apakah komputer menang atau tidak.

void CheckWin (char \* state, int \* win) {

FILE \*fp;

char MARK[9], s[5];

fp = fopen("file.txt", "r");

fscanf(fp, "%s", MARK);

while (strcmp(MARK, "MENANG") != 0) {

fscanf(fp, "%s", MARK);

}

fscanf(fp, "%s", s);

while (strcmp(s, "END") != 0) {

if (strcmp(s, state) == 0) {

\*win = 1;

break;

} else {

fscanf(fp, "%s", s);

}

}

}

* 1. **Prosedur CheckTie**

Prosedur CheckTie adalah fungsi yang digunakan untuk membaca daftar *state* seri pada file eksternal dan menetukan apakah *state* saat ini merupakan *state* di mana terjadi seri. Prosedur ini memiliki parameter CState dan tie, di mana CState adalah *state* permainan pada saat ini dan tie adalah “boolean” yang menyatakan apakah permainan seri atau tidak.

void CheckTie (char \* state, int \* tie) {

FILE \*fp;

char MARK[9], s[5];

fp = fopen("file.txt", "r");

fscanf(fp, "%s", MARK);

while (strcmp(MARK, "SERI") != 0) {

fscanf(fp, "%s", MARK);

}

fscanf(fp, "%s", s);

while (strcmp(s, "END") != 0) {

if (strcmp(s, state) == 0) {

\*tie = 1;

break;

} else {

fscanf(fp, "%s", s);

}

}

}

* 1. **Prosedur Play**

Prosedur Play adalah prosedur yang menggabungkan fungsi Print, Trans, CheckWin, dan CheckTie. Prosedur ini tidak memiliki parameter.

void play () {

char CState[5] = "q000";

char history[100] = "q000";

char first, dump[100];

int move;

int win, tie = 0;

printf("\nFirst player (x = you, o = CPU): ");

scanf(" %c", &first);

while ((first != 'x') && (first != 'o')) {

printf("Your input is not 'x' or 'o'.\n");

printf("Re-input: "); scanf(" %c", &first);

}

printf("\nFirst move is filling the middle slot.\n");

if (first == 'x') {

strcpy(CState, Trans(CState, -1));

}

if (first == 'o') {

strcpy(CState, Trans(CState, 0));

}

Print(CState);

strcat(history, " -> "); strcat(history, CState);

while ((win == 0) && (tie == 0)) {

printf("Your next move (1-9): ");

if (scanf("%d", &move)) {

if ((move >= 1) && (move <= 9)) {

strcpy(CState, Trans(CState, move));

printf("\n"); Print(CState);

strcat(history, " -> "); strcat(history, CState);

CheckWin(CState, &win);

CheckTie(CState, &tie);

} else {

printf("You can only input numbers 1-9.\n\n");

}

} else {

printf("You can only input a number!\n\n");

scanf("%s", dump);

}

}

if (win) {

printf("\nCPU wins.\n");

} else if (tie) {

printf("\nIt's a tie.\n");

} else {

printf("\nDefintely an error...\n");

}

printf("\nStates history: %s\n", history);

}

1. **CONTOH INPUT-OUTPUT**

Untuk memastikan program berjalan dengan baik, dilakukan beberapa pengujian. Berikut merupakan contoh input dan output program. Output ditebalkan dan diberi garis bawah.

***Tes 1:***

First player (x = you, o = CPU): **y**

Your input is not 'x' or 'o'.

Re-input: **a**

Your input is not 'x' or 'o'. Also, I can only read one word at a time. Please don't tease me.

Re-input: **x**

First move is filling the middle slot.

o - -

- x -

- - -

Your next move (1-9): **7**

o - o

- x -

x - -

Your next move (1-9): **2**

o x o

- x -

x o -

Your next move (1-9): **4**

o x o

x x o

x o x

It's a tie.

States history: q000 -> q126 -> q131 -> q137 -> q166

***Test 2:***

First player (x = you, o = CPU): **o**

First move is filling the middle slot.

- - -

- o -

- - -

Your next move (1-9): **0**

You inserted 0. You can only input numbers 1-9.

Your next move (1-9): **a**

You can only input numbers!

Your next move (1-9): **3**

- - x

- o -

o - -

Your next move (1-9): **2**

o x x

- o -

o - -

Your next move (1-9): **4**

o x x

x o -

o - o

CPU wins.

States history: q000 -> q001 -> q004 -> q017 -> q070

1. **REFERENSI**

Subramaniam, Arjun. (2015). *Is there a way to never lose at Tic-Tac-Toe?*. Diakses

September 2018, dari

<https://www.quora.com/Is-there-a-way-to-never-lose-at-Tic-Tac-Toe>.

Wikipedia. (2018). *Tic-tac-toe*. Diakses September 2018, dari

<https://en.wikipedia.org/wiki/Tic-tac-toe>.