LAPORAN TUGAS BESAR

IF2111/Algoritma dan Struktur Data STI

Catur

Dipersiapkan oleh:

Nyoman Kevin Cahyadi Giri / 18218001

Adrian Timotheus Salim / 18218002

Adi Hendro / 18218009

Vincentius Ian Widi Nugroho / 18218034

Sistem dan Teknologi Informasi

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

Daftar Isi

1 Ringkasan

Pada tugas besar kali ini, kami membuat permainan catur dengan memanfaatkan ADT array, stack, queue, list linier, mesin karakter dan mesin kata. Papan catur berukuran 8x8 kotak dan dibuat dengan memanfaatkan adt array untuk menyimpan bidak-bidak yang masih ada di papan. Kondisi awal bidak diposisikan seperti permain catur pada umumnya di mana player 1(putih) berada di bawah papan dan player 2(hitam) berada di atas papan.

Pada awal permainan, pemain diberi 3 pilihan. 'New game' untuk memulai permainan baru, 'load game' untuk melanjutkan permainan sebelumnya, dan 'leaderboard' untuk melihat high score dari permainan sebelumnya dengan format 'nama – skor' diurutkan dari skor terbesar dan diurutkan berdasarkan abjad jika skor sama. Fungsi leaderboard dan load game menggunakan ADT mesin kata untuk membaca file external .txt. Jika pemain memilih 'new game', pemain menginput 3 karakter abjad untuk dipakai di high score game. Kemudian, permainan dimulai.

Terdapat 6 jenis bidak berbeda sesuai permainan catur pada umumnya. Tiap jenis memiliki poin dan arah gerak yang berbeda. Perlu diperhatikan bahwa bidak tidak bisa bergerak jika ada bidak sejenis yang menghalangi (kecuali kuda). Bidak tentunya juga bisa memakan bidak lawan. Daftar bidak milik pemain dan lawan disimpan dalam sebuah list linier. Ketika sebuah bidak termakan, bidak tersebut akan dihapus dari list untuk dihitung jumlah akumulasi poin milik lawan. Pemain juga bisa menggunakan gerakan 'castling' dan 'en passant'. Untuk pion, jika pion sudah mencapai ujung papan dapat dipromosikan menjadi kuda, benteng, menteri, atau ratu sesuai keinginan pemain.

Ketika seorang pemain memulai gilirannya, pemain memiliki 4 opsi command. 'MOVE', 'SPECIAL MOVE', 'UNDO', dan 'SAVE'. 'MOVE' digunakan untuk menggerakkan bidak sesuai arah gerak bidaknya. Ketika command ini dijalankan, akan muncul daftar pilihan bidak yang bisa dijalankan, lalu player menginput angka untuk memilih bidak yang ingin dijalankan. Setelah itu, akan muncul daftar daerah yang bisa menjadi tujuan bidak tersebut. Promosi dilakukan secara otomatis pada saat move. 'SPECIAL MOVE' mencakup 'castling' dan 'en passant' dan hanya muncul jika ada bidak yang dapat melakukan salah satu dari kedua special move tersebut. 'UNDO' dilakukan untuk membatalkan gerakan kita dan gerakan lawan. Misal kita melakukan suatu move dan ingin undo, kita menggunakan giliran lawan untuk menginput undo setelah itu giliran kita lagi. Daftar gerakan disimpan pada ADT stack agar jika undo dijalankan, dapat diambil lagi dengan mudah. 'SAVE' digunakan untuk menyimpan keadaan papan sekarang dan disimpan dalam file eksternal.

Permainan diawali oleh player 1(putih) lalu dilakukan bergantian. Giliran pemain disimpan pada ADT queue. Papan akan selalu muncul tiap giliran pemain dan tiap pemain melakukan suatu gerakan, papan tersebut akan diupdate. Permainan berakhir jika pemain sudah melakukan 50 langkah untuk tiap pemain (tidak termasuk undo). Selain itu permainan bisa berakhir jika raja termakan atau terskakmat. Jika raja lawan berhasil diskakmat, pemain mendapat tambahan 20 poin. Setelah permainan berakhir, ditampilkan skor masing-masing pemain untuk dimasukkan dalam leaderboard. Hanya score pemain yang tertinggi yang disimpan dalam leaderboard. Jika kedua skor sama, tidak ada penyimpanan high score.

Dalam laporan ini dijelaskan bagaimana implementasi kode yang digunakan untuk menjalankan spesifikasi di atas. Lengkap dengan penjelasan struktur data, isi program utama,

dan algoritma yang kami anggap menarik. Dicantumkan pula daftar test apa saja yang kami lakukan beserta lampiran.

Kesimpulan setelah mengerjakan tugas besar ini adalah tugas ini sudah cukup memenuhi kriteria wajib yang diperlukan. Namun, permainan catur ini masih belum sempurna dan masih memiliki kekurangan. Program ini masih belum menyerupai permainan catur sesungguhnya seperti masih bisa memakan raja dan syarat skakmat masih belum jelas. Ke depannya, program ini masih bisa dikembangkan.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 Variasi Command Undo

Ketika command 'UNDO' dijalankan, seharusnya gerakan kedua terakhir akan dibatalkan seusai spesifikasi tugas. Dengan kata lain, ada 2 gerakan yang diambil dari stack gerakan. Namun, jika pemain baru menggerakkan bidak satu kali, hanya ada 1 gerakan dalam stack. Jadi, untuk kasus giliran pertama, program hanya akan mengambil 1 gerakan dari stack dan bukan 2 seperti biasanya.

Selain itu, fungsi undo harus bisa membatalkan promosi dengan menambahkan elemen boolean promoted pada pion. Jika terpromosi, pion terpromosi yang masuk ke stack promotednya menjadi true. Fungsi undo juga bisa mengundo bidak yang termakan dengan melihat posisi stack termakan.

2.2 Menginput String dan Mengubahnya Menjadi Integer

Di dalam program, terdapat pilihan untuk menginput suatu bilangan integer untuk melakukan command. Contohnya, di awal program, input 1 untuk New Game, 2 untuk Load Game, dan 3 untuk melihat leaderboard. Namun, program hanya menerima bilangan integer, jadi, jika kita menginput char, program akan error. Karena itu, program dibuat function agar menerima strings. Setelah itu, function tersebut merubah semua char dalam strings tersebut menjadi bilangan ascii yang kemudian diubah menjadi integer lagi. Function ini pun mereturn integer ascii ini, jadi program besar membaca integer ascii dari char.

2.3 Menambahkan Warna pada Tampilan Bidak

Untuk memperindah tampilan bidak, kami membedakan bidak kawan dan bidak lawan dengan menambahkan warna. Bidak player 1 berwarna hijau dan bidak player 2 berwarna merah. Ditambahkan juga beberapa warna pada program catur ini agar tampilan menjadi lebih menarik. ^_^

2.4 Save dan Load

Fungsi save secara singkat berarti memindahkan elemen-elemen dari ADT ke file .txt. Yang kami gunakan adalah list_ada_hitam.txt, list_ada_putih.txt, history.txt, yang membaca dari list linear hitam & putih dan mencatat nama, player, poin, danposisi bidak. Termakan.txt mencatat

data bidak yang termakan. namaplayer.txt mencatat nama player. poin.txt membaca total poin hitam dan putih, serta giliran paling baru dan turnnya. leaderboard.txt mencatat semua highscore yang pernah didapat. Fungsi load adalah kebalikan dari save. Di mana program membaca semua file .txt di atas untuk dijadikan elemen dalam list, stack, dll.

2.5 En Passant

Kami membuat 2 fungsi, cekEnPassant untuk mengecek bisa/tidaknya en passant dan void enpassant untuk melakukannya. cekEnPassant melihat stack history untuk melihat apakah ada pion yang baru berjalan 2 langkah. Jika ada, list pemain aktif akan dicek apakah ada pion yang letaknya bersebelahan dengan pion yang baru bergerak dua langkah tersebut. Jika ada, cekEnPassant = true. Void EnPassant memindahkan pion kita ke belakang pion lawan dan memasukkan pion yang termakan ke stack termakan.

3 Struktur Data (ADT)

3.1 Piece

Tipe piece dipakai untuk menyimpan data-data dari tiap bidak yang ada. Tipe piece terdiri dari:

```
typedef struct {
                    char nama;
                    int player;
                    int poin;
                    int posisiR;
                    int posisiC;
             } piece;
```

'nama' berisi nama-nama piece, direpresentasikan dengan 1 char. Contoh: pion='C', Raja='K', Queen='Q'. Ini digunakan untuk membedakan bidak-bidak untuk menentukan arah geraknya dan penempatannya di papan. 'player' berisi integer 1 atau 2. 1 untuk putih dan 2 untuk hitam. Digunakan untuk dicocokkan dengan queue giliran. 'poin' berisi jumlah poin dari tiap bidak, sesuai dengan nama bidaknya. Jika bidak termakan, kita menghitung akumulasi poin ini. posisiR berisi posisi baris bidak sesuai dengan array. Jika bidak di atas, posisiR=1, jika bidak di bawah, posisiR=8. posisiC berisi posisi kolom bidak. Jika bidak di kiri, posisiC=1, jika di kanan = 8.

3.2 Papan

Kami membuat tipe bentukan papan yang berisi:

```
typedef struct {
                    char nama;
                    int player;
             } papan;
```

Jika tipe piece digunakan untuk melihat bidak secara individual, tipe papan dipakai untuk melihat seluruh bidak pada papan catur. Dengan kata lain papan catur kami adalah sebuah array of papan. Papan catur kami adalah array 10x10 dengan indeks 0-9. Indeks 0 dan 9 digunakan untuk membuat border dengan nama='*'. Guna border ini adalah jika ada bidak di ujung papan, bisa mengecek sekeliling kotaknya tanpa error. Contoh ada bidak di array indeks 8,8 dan akan

dicek apakah daerah di sekeliling bidak bisa dipenuhi. Ketika program melihat indeks 9,9 program tidak akan melihat keluar array.

3.3 ADT List

Dalam program, ADT list dipakai untuk list bernama list_ada_putih, list_ada_hitam, list_bisa_gerak, list_posisi. Tipe data dari list adalah sebagai berikut:

```
typedef struct {
            infotype info;
            address next;
        } list;
```

List_ada_putih dan list_ada_hitam, sesuai namanya berfungsi untuk menyimpan bidak putih atau bidak hitam yang masih ada di papan. Penggunaan list_ada_putih atau list_ada_hitam tergantung pada giliran, jadi kedua list ini tidak akan dipakai bersamaan dalam sekali giliran.
Selanjutnya, list_bisa_gerak berisi bidak mana saja dari list putih/hitam yang bisa bergerak. Kami memanfaatkan fungsi bernana cek_bisa_gerak yang merupakan bagian dari prosedur move. Fungsi ini mengecek secara traversal dari list hitam/putih dan mengecek bidak mana yang bisa bergerak. Fungsi ini tidak mengecek ke mana saja gerakan bidak yang bisa dituju, hanya bisa/tidaknya bergerak, termasuk memakan. Jika fungsi menghasilkan true, maka bidak dalam list dimasukkan ke list_bisa_gerak. Ketika menjalankan fungsi move, program harus bisa mengeprint daftar bidak yang bisa bergerak. Daftar bidak yang diprint tersebut dibaca dari list_bisa_gerak.
Terakhir, adalah list_posisi. List ini merupakan tujuan berikutnya dari list_bisa_gerak. Ketika player menginput pilihan n, program akan memilih bidak ke-n dari list_posisi dan memasukkan bidak tersebut ke fungsi bernama cek_semua_gerak. Fungsi ini berfungsi untuk memasukkan daftar tujuan suatu bidak yang mungkin dilakukan ke list_posisi. Program kemudian mengeprint list_posisi ini agar bisa ditampilkan dan dipilih player. List ini berisi tipe bentukan 'posisi' yang berisi posisiC dan posisiR diambil dari bidak.

3.4 ADT Queue

Untuk ADT Queue, kami menggunakan bentukan yang sama yang diberikan saat kuliah.

```
typedef struct {
            infotype_queue * T;
                    address HEAD;
                  address TAIL;
                   int maxEl;
               } queue;
```

Isi infotype_queue berisi integer 1 dan 2. Program melihat elemen head dari queue untuk melihat giliran. Jika giliran sudah dijalankan, queue akan menghapus nilai head dan menaruhnya pada tail.

3.5 ADT Stack

ADT Stack memiliki struktur sebagai berikut:

```
typedef struct {
    infotype_stack T[MaxEl_stack+1];
    address_stack TOP;
} stack;
```

```
Stack yang kami buat memuat infotype_stack yang terbentuk dari:
typedef struct {
          char nama;
          int player;
          int poin;
          int turn;
          int posisiR_lama;
          int posisiC_lama;
          int posisiR_baru;
          int posisiC_baru;
          boolean promotion;
          boolean twosteps;
} infotype_stack;
```

Stack berfungsi untuk mewujudkan fungsi undo. Ada 2 stack di program kami, stack history dan stack termakan. Tiap kali sebuah bidak digerakkan, dalam arti berpindah tempat atau memakan/termakan, data bidak tersebut akan dimasukkan ke stack history. Data utamanya adalah nama bidak, nomor player (1 atau 2), poin, posisi lama dan posisi baru. Dalam permainan biasa, jika seseorang melakukan undo, dia membatalkan gerakan terakhirnya dan gerakan terakhir lawannya. Itu berarti, program mengambil 2 elemen dari stack untuk mengembalikan bidak ke posisi lama yang sudah tercatat di infotype_stack.
Ada 3 bagian infotype yang sedikit istimewa, yaitu turn, promotion, dan twosteps. Turn mencatat berapa kali bidak sudah bergerak. Jika bidak belum pernah digerakkan(kondisi awal permainan), maka turn=0 sehingga tidak ada yang bisa diundo. Jika baru 1 player yang bergerak, turn=1 sehingga alih-alih membatalkan gerakan kedua player, hanya 1 player yang dibatalkan gerakannya. Boolean promotion digunakan untuk mengundo pion yang promoted agar fungsi promotion bisa diundo. Jika dalam stack terdapat ratu dan promotion=true, undo mengembalikannya menjadi pion lagi dan bukan ratu. Lalu ada boolean twosteps untuk melihat kebisaan EnPassant. Jika pion lawan baru bergerak 2 langkah, twosteps=true dan kita bisa EnPassant.

3.6 ADT Mesin Kata dan Mesin Karakter

Mesih Karakter dan Mesin Kata mengikuti ADT kuliah hanya saja marknya adalah ';' dan dalam file .txt, tiap karakter dibatasi oleh '|'. ADT Mesin Kata dan Karakter dipakai untuk save dan load seperti dijelaskan pada poin 2.4.

4 Program Utama

Untuk bagian include, ADT yang diimplementasikan kembali adalah stack.c, listlinier.c, queue.c. Header yang digunakan adalah tipe_bentukan.h. Untuk subprogram ada start.c, inisialisasi.c, print_papan.c, listposisi.c, ascii_checker.c, move.c, special_move.c, undo.c, save.c, load.c, urut_leaderboards.c

Saat game dimulai, pemain diberikan pilihan untuk melakukan new game, load game, ataupun menampilkan leaderboard. Pemain kemudian diminta untuk menginput suatu integer dari 1-3. Jika pemain menginput 1, yaitu new game, pemain akan diminta untuk memasukkan nama pemain beserta lawan pemain. Bersamaan dengan input nama, program akan menampilkan papan yang sudah diinisialisasi di fungsi inisialisasi.c. Setelah itu, nama pemain akan disimpan

dalam array pemain hitam dan putih. Jika pemain menginput 2, yaitu load game, maka program akan load posisi catur yang sebelumnya yang telah disimpan melalui subprogram save. Jika pemain menginput 3, maka leaderboard akan ditampilkan maksimal 10 skor teratas. Leaderboard akan ditampilkan hingga pemain menekan tombol Y atau y. Setelah pemain menekan tombol y atau Y, maka kembali ke game mulai.

Selama permainan berjalan, pemain dapat memasukkan command MOVE, SPECIAL_MOVE, UNDO, SAVE, atau RESET. Jika pemain menginput MOVE, maka program akan menampilkan daftar pilihan pion ataupun perwira yang dapat bergerak melalui subprogram cek_bisa_gerak.c. Ketika pemain menginput pilihan bidak yang bersesuaian dengan nomornya, maka program akan memanggil subprogram cek_semua_gerak.c yang akan menampilkan semua pilihan posisi yang mungkin. Setelah itu, giliran akan berganti. Jika pemain menginput SPECIAL_MOVE, maka program akan memanggil fungsi special_move.c yang menampilkan daftar special move apa saja yang dapat dilakukan, bisa berupa castling, promotion, ataupun en passant. Jika pemain menginput nomor yang bersesuaian, maka special_move akan dilakukan. Jika pemain menginput UNDO, maka pemain akan mundur 1 langkah per pemain, dengan giliran sekarang adalah giliran pemain yang mengetikkan undo. Jika pemain menginput SAVE, maka posisi akan disave melalui subprogram save.c. Jika pemain menginput RESET, maka posisi akan dikembalikan menjadi posisi awal jika user mengkonfirmasi dengan mengetikkan Y/y.

Permainan akan berakhir jika telah mencapai skakmat, stalemate, ataupun 50 langkah telah dilakukan oleh masing" pemain. Kemenangan akan ditentukan dengan peraih poin tertinggi. Apabila poin yang diraih lebih tinggi dari poin yang ada pada leaderboard, maka poin yang diraih akan dimasukkan ke leaderboard. Dan poin terkecil akan dihapus dari list.

5 Algoritma-Algoritma Menarik

5.1 Algoritma gerak_aman

```
void gerakaman(list kawan, list lawan, papan *board[10][10], list *list_bisa_gerak,
boolean *endgame){
    address_list P, R, R1;
    address_posisi Q, PrecQ;
    // lihat bidak yang masih ada di papan dari list linier dan memasukkan ke
list_bisa_gerak
    P = First(kawan);
    CreateEmpty_list(list_bisa_gerak);
    while (P != Nil_list) {
        if (cekbisagerak(Info(P), board)) { //Info(P) == piece
            InsVFirst(list_bisa_gerak, Info(P)); //dari list kawan dimasukkin ke
list_bisa_gerak

            R = First(*list_bisa_gerak); //list_linier yang bisa gerak dari kawan

            CreateEmpty_posisi(&Gerakan(R));
```

```
            ceksemuagerak(Info(R), board, &Gerakan(R));

            Q = First(Gerakan(R)); //list_posisi gerakan yang bisa dilakukan piece
kawan
            PrecQ = Q;

            while (Q != Nil_list){ //cek semua gerakan bidak tersebut, bikin skak atau
tidak
                if(jadi_skak(lawan, kawan, board, Info(R), Info(Q).posisiR,
Info(Q).posisiC)){

                    if(Q == First(Gerakan(R))){ //jika elemen gerakan pertama
                        DelFirst_posisi(&Gerakan(R),&Q);
                    } else{ //bukan elemen gerakan pertama
                        DelAfter_posisi(&Gerakan(R),&Q,PrecQ);
                    }
                }
                PrecQ = Q;
                Q = Next(Q);
            }

            if(IsEmpty_posisi(Gerakan(R))){ //kalo bidak tersebut ga bisa gerak karena
bikin skak
                DelFirst(list_bisa_gerak, &R1); //maka didelete
            }
        }
        P = Next(P);
    }
    if(IsEmpty_list(*list_bisa_gerak)){ //kalo ga ada bidak yg bisa gerak
        //berarti STALEMATE atau CHECKMATE
        *endgame = true;
    } else{
        *endgame = false;
    }
}
```

Pada permainan catur yang sesungguhnya, Raja dan pergerakannya sangatlah spesial. Raja tidak bisa dimakan dan tidak bisa pindah ke kotak yang menyebabkan dirinya terancam. Selain itu, ketika Raja sedang diskak, pemain hanya bisa menggerakkan Raja dan bidak lain yang bisa menghalangi bidak musuh penyekak. Ia tidak bisa menggerakkan bidak yang tidak berhubungan dengan keselamatan Raja.

Ketika Raja terskak, tidak bisa bergerak lagi, dan tidak ada bidak yang bisa menolong, maka kondisi tersebut ialah skakmat. Jika Raja tidak bisa bergerak dan tidak ada bidak lain yang bisa digerakkan, namun Raja tidak sedang dalam kondisi terskak, maka kondisi tersebut disebut stalemate.

Algoritma gerak_aman ini dipakai di fungsi main.c sebelum pemain memilih gerakan yang diinginkan.

Algoritma gerak_aman memiliki sebuah parameter output berupa list yang bernama list_bisa_gerak. Gerak_aman akan mencari semua bidak yang bisa digerakkan dan dicatat di list_bisa_gerak. Lalu, di dalam list_bisa_gerak dibuat lagi sebuah list posisi yang mencatat semua gerakan dari bidak tersebut. Setiap gerakan yang ada akan dicek dengan fungsi jadi_skak. Jika bidak tersebut digerakkan ke posisi tadi dan menyebabkan Raja sendiri skak, maka gerakan ke posisi tadi akan dihapus. Jika bidak tersebut tidak bisa gerak sama sekali, maka bidak tersebut akan dihapus dari list_bisa_gerak. Di akhir jika list_bisa_gerak ternyata kosong, berarti terjadi kondisi skakmat ataupun stalemate, yang akan dicek lagi menggunakan algoritma is_skak.
Dengan demikian, Raja tidak bisa dimakan, Raja hanya dapat bergerak ke kotak yang aman, dan bidak lain harus menyelamatkan Raja jika Raja sedang diskak.
Algoritma ini menyebabkan permainan catur kami menjadi permainan catur yang sesungguhnya yang selalu menjaga Raja.

5.2 Algoritma is_skak

```
boolean isskak(list lawan, list kawan, papan *board[10][10], address_list *K,
address_list *P1){
    *K = First(kawan);
    while ((*K!=Nil_list) && (Info(*K).nama!='K'))
        *K = Next(*K);

    return (isthreaten(lawan, Info(*K).posisiC, Info(*K).posisiR, board, P1));
}

boolean isthreaten(list lawan, int x, int y, papan *board[10][10], address_list *P1){
    //mengefound apakah suatu spot berbahaya untuk raja
    list_posisi dftr_posisi;
    address_posisi R;
    address_list P;
    boolean found = false;
    P = First(lawan);

    while (P!=Nil_list){
        CreateEmpty_posisi(&dftr_posisi);
        ceksemuagerak(Info(P), board, &dftr_posisi); //menyimpan semua gerakan yang
mungkin dari bidak lawan
        R = First(dftr_posisi);

        while (R!=Nil_list){
            if((Info(R).posisiC==x)&&(Info(R).posisiR==y)){
                found = true;
                *P1 = P;
            }
            R = Next(R);
        }
        P = Next(P);
```

```
    }
    return found;
}
```

Algoritma is_skak akan mengecek apakah suatu kotak di papan berbahaya jika dikunjungi oleh Raja. Algoritma ini dipakai di fungsi gerak_aman di dalam main.
Algoritma is_skak mengecek semua bidak lawan dan gerakannya. Jika ada satu saja bidak lawan yang bisa pindah menuju ke sebuah kotak, maka kotak tersebut akan dianggap tidak aman bagi Raja. Algoritma ini sangat berguna dalam membuat list_bisa_gerak yang aman dan tidak berbahaya bagi Raja.

5.3 Algoritma delay

```
void delay(int milliseconds){
    long pause;
    clock_t now,then;

    pause = milliseconds*(CLOCKS_PER_SEC/1000);
    now = then = clock();
    while((now-then) < pause)
        now = clock();
}
```

Dengan memakai header time.h, algoritma delay ini dapat membuat program berhenti untuk sementara waktu. Satuan yang dipakai ialah milidetik. Algoritma tersebut dipakai di dalam fungsi start yang menampilkan menu utama dan di dalam fungsi main. Setiap kali pemain selesai melakukan gerakan, program akan delay selama 1 sampai 1,5 detik agar status pergerakan bidak dapat terbaca dengan baik.

5.4 Algoritma ascii_checker

```
void string2ByteArray(char* input, int* output)
{
    // mengubah byte string ke array of integer
    int loop = 0;
    int i = 0;

    while(input[loop] != '\0')
        output[i++] = input[loop++];
}
void stringToInt(char* ascii_str,int* Ret){
    //mendapatkan nilai bilangan integer pertama dari array of string
    int len = strlen(ascii_str);
    int arr[len];
    string2ByteArray(ascii_str, arr);
    *Ret = arr[0];
```

```
}

do{
        choice = (char*) malloc (sizeof(100));
        printf("Your choice: ");
        scanf("%s",choice);
        stringToInt(choice,&X);
        if(!(X==49 || X==50 || X==51))//49 adalah ascii untuk 1, 50 = 2, 51 = 3
            printf("Wrong input! It's must be a number from 1-3!\n\n");
    } while(!(X==49 || X==50 || X==51));
```

Dengan memakai header string.h, algoritma ini dapat mengecek format ASCII dari inputan user. Dipakai di banyak fungsi, seperti di start, main, move, special_move. Ketika pemain memasukkan angka atau karakter di luar yang sudah ditetapkan, program akan meminta user untuk memasukkan input kembali hingga benar. Dengan demikian bisa diminimalisasi kesalahan pengetikan input.

6 Data Test

6.1 Main.c

MOVE -> bidak atau perwira dapat bergerak ataupun memakan, ganti giliran
Hasil: dapat mendeteksi bidak mana yang bisa digerakkan, memunculkan gerakan yang dapat dilakukan, dan menggerakkan bidak tersebut
Data test: menggerakkan pion, menggerakkan kuda
SPECIAL_MOVE -> pemain dapat melakukan enpassant, castling, ataupun promotion
Hasil: dapat mendeteksi dan melakukan castling, mendeteksi dan melakukan enpassant, terjadi promotion jika pion sampai akhir.
Data Test: bidak yang akan castling, bidak yang akan enpassant, bidak yang akan promotion

6.2 Stack.c

Undo -> membatalkan gerakan
Hasil: membatalkan gerakan pemain dan gerakan lawan
Hasil: membatalkan gerakan pemain saja apabila baru bergerak sekali.
Data Test: mengetikkan UNDO sebagai command.

6.3 Print_papan.c

Print_papan -> menampilkan gambar papan catur
Hasil: papan catur yang indah berhasil ditampilkan melalui command line interface
Data test: memanggil fungsi print_papan

6.4 Ada_orang.c

Ada_orang -> mengecek apakah 1 kotak kosong apabila ditempati

Hasil: mengembalikan true/false tergantung isi kotak
Data test: Mengisi 1 square dengan bidak/perwira melalui fungsi construct, kemudian memangggil fungsi ada_orang.c dengan parameter posisi X dan Y kotak tersebut.

6.5 Ascii_checker.c

Ascii_checker -> mengconvert sebuah karakter ataupun bilangan menjadi kode ASCII
Hasil: bilangan atau karakter yang dimasukkan berubah menjadi kode ASCII
Data test: memanggil fungsi ascii_checker dengan parameter sembarang karakter/bilangan, kemudian print hasilnya

6.6 Queue.c

Queue -> mengecek giliran serta mengganti giliran jika move sudah selesai
Hasil : giliran berhasil diganti
Data test: queue giliran

7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Move	Menggerakkan bidak	Menjalankan move, memilih pion b2 dan memindahkannya ke b4	Pion b2	Pion pindah ke b4	Pion pindah ke b4
2	Move	Menggerakkan bidak jika terhalang	Menggerakkan pion c2 ke c3 lalu mencoba menggerakkan kuda b1	Kuda b1	Kuda hanya bisa bergerak ke a3 melewati pion b2	Sesuai dengan yang diharapkan
3	Castling	Melakukan castling jika syarat terpenuhi	Mengosongkan f1,g1, lalu melakukan castling antara e1 dan h1	Memilih castling	Raja dan rook berhasil melakukan castling	Sesuai yang diharapkan
4	En Passant	Melakukan En Passant jika syarat terpenuhi	Menjalankan pion a2 sampai a5 lalu menggerakkan pion b7 ke b5 lalu melakukan En Passant	Pilih En Passant	Pion berhasil melakukan En Passant	Sesuai yang diharapkan
5	Promotion	Melakukan promotion jika pion sudah berada di ujung	Menggerakkan pion sampai ke ujung papan	Memilih pion lalu memilih ingin promote menjadi apa	Pion berhasil melakukan promotion	Sesuai yang diharapkan
6	Array	Print isi array menjadi papan	Mengisi array dengan membaca	Array yang berisi daftar bidak	Papan posisi bidak dan warna bidak	Jika dijalankan via driver, warna tidak

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		yang lebih terlihat	list Print isi array		diprint dengan benar	muncul
7	Queue	Mengecek giliran serta mengganti giliran jika move sudah selesai	Menjalankan move Mengisi queue dengan 1 atau 2 tergantung info(Tail).	Queue giliran	Bisa mengganti giliran dari 1 (putih) menjadi 2 (hitam)	Giliran berhasil diganti dari 1 (putih) menjadi 2 (hitam)
8	Stack	Mengecek apakah fungsi pada stack.c berhasil	CreateEmpty(Stack) dan mengeceknya apakah stack benar-benar kosong dengan menggunakan fungsi IsEmpty(Stack). Mencoba untuk menambah (Push) dan mengeluarkan (Pop) stack	Masukan_stack P,1,1,3,7,2,5,2,0,1,0	keluaran_stack P,1,1,3,7,2,5,2,0,1,0	keluaran_stack P,1,1,3,7,2,5,2,0,1,0
9.	Leaderboards	Menampilkan leaderboard yang berisi maksimal 10 skor teratas yang sudah terurut mengecil berdasarkan skor dan terurut membesar berdasarkan nama apabila skor sama.	Memasukkan nama dan skor baru ke leaderboard yang akan langsung terurut.	Input KEV – 40 Data yang sudah ada DRI – 50 ADI – 10 IAN - 5	Output DRI – 50 KEV – 40 ADI – 10 IAN – 5	Output DRI – 50 KEV – 40 ADI – 10 IAN – 5
10.	List	Mengecek apakah construct bidak pada list_ada_hitam dan list_ada_putih berhasil	Membuat list kosong Construct bidak Ngecek list kosong atau tidak Print isi list, invers list	Sebagian bidak catur	Bidak catur dapat di insert ke dalam list	Nama bidak catur, pemain dan poinnya ditampilkan
11.	Mesin Karakter dan Kata	Mengecek apakah fungsi-fungsi pada ADT Mesin Kata dan Karakter berfungsi	Membuka sebuah file eksternal dan membaca kata-kata yang ada di dalamnya. Membaca dengan menggunakan ADV() dan ADVKATA()	Data file eksternal : kevin|adi|ian|adrian	Dengan ADV() kevin|adi|ian|adrian Dengan ADVKATA() Nama orang ke-1: kevin Nama orang ke-2: adi Nama orang ke-3: Ian Nama orang ke-4: adrian	Dengan ADV() kevin|adi|ian|adrian Dengan ADVKATA() Nama orang ke-1: kevin Nama orang ke-2: adi Nama orang ke-3: Ian Nama orang ke-4: adrian
12.	ASCII	Menerima	Input dimasukkan	1	49	49

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	Checker	karakter/bilangan dan mengubahnya menjadi nilai ASCII yang bersesuaian	Input nanti dikonversi ke kode ASCII Kode ASCII di cek apakah sesuai dengan range input	2 3	50 51	50 51

Tabel 7.1 : Tabel Test Case

8 Pembagian Kerja dalam Kelompok

Nama	NIM	Pembagian Kerja
Nyoman Kevin Cahyadi Giri	18218001	- Save - Load - Move - Leaderboard - Queue - Implementasi List - Laporan
Adrian Timotheus	18218002	- Fungsi cek_semua_gerak - Castling - Promotion - Laporan
Adi Hendro	18218009	- Fungsi cek_semua_gerak - Castling - Promotion - En Passant - Fungsi cek_skakmat - Stalemate - Implementasi List - Undo - Laporan
Vincentius Ian	18218034	- Fungsi print papan - Fungsi cek_bisa_gerak - Fungsi cek_skak - Fungsi cek_skakmat - Laporan

Tabel 8.1 : Pembagian Kerja Antar Anggota Kelompok

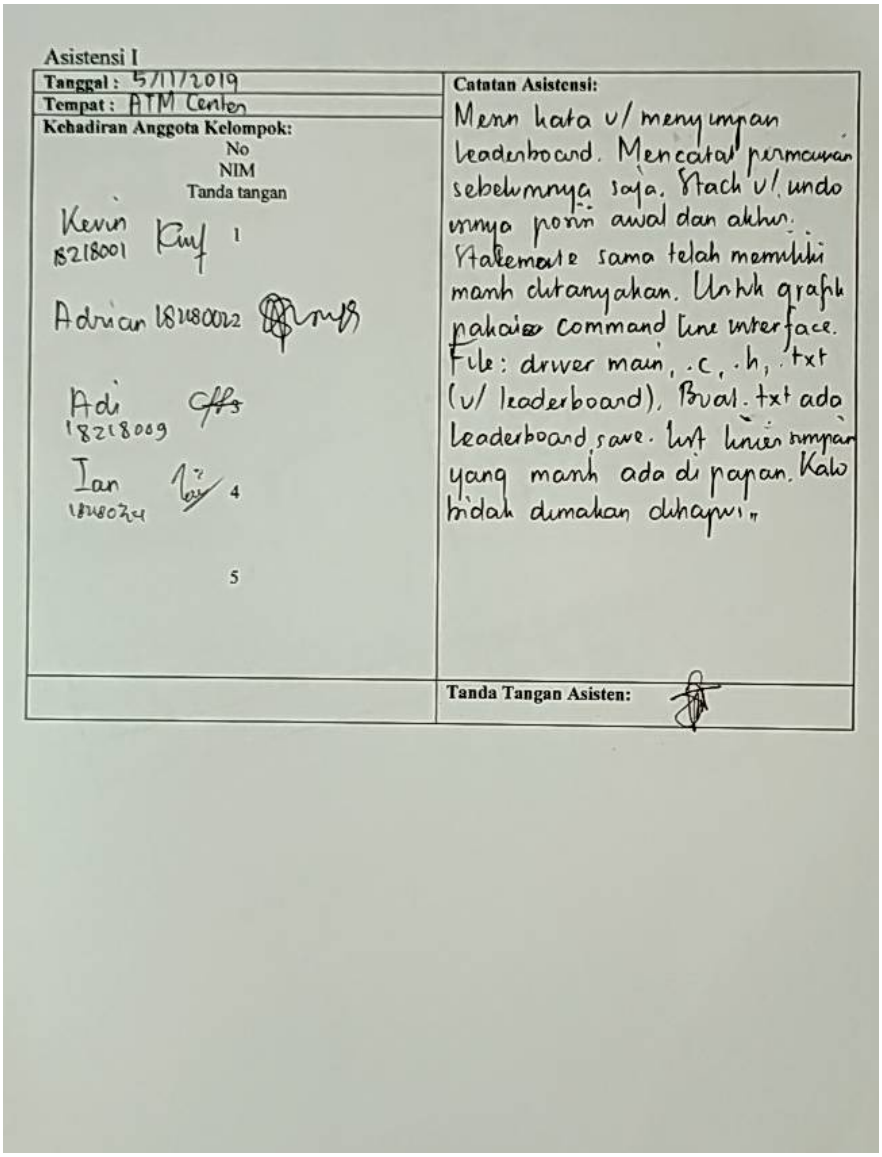
9 Lampiran

9.1 Deskripsi Tugas Besar

Membuat game catur yang menggunakan ADT array untuk menampilkan papan dan bidak yang dimiliki kedua pemain, ADT stack untuk menyimpan gerakan bidak untuk kedua pemain, ADT

queue untuk menampung giliran pemain, ADT list linier untuk menampung bidak yang dimiliki pemain besertas posisi, serta mesin karakter.untuk membaca file external .txt dan leaderboard, serta ADT lain sesuai kebutuhan. Game catur diawali dengan main menu dengan pilihan new game, load game, serta leaderboard. Papan catur ditampilkan dengan menggunakan command line interface. Bidak catur dan gerakannya mengikuti aslinya, termasuk promosi, en passant, serta castling. Pemain menjalankan bidak dengan menggunakan command MOVE, SPECIAL MOVE, ataupun menjalankan fungsi lainnya dengan UNDO ataupun SAVE. Permainan berakhir apabila kedua pemain telah memainkan 50 langkah atau skakmat (mana yang tercapai terlebih dahulu).

9.2 Notulen Rapat



Asistensi I

Tanggal : 5/11/2019
Tempat : ATM Center
Kehadiran Anggota Kelompok:
No
NIM
Tanda tangan

1 Kevin 18218001

Adrian 18218002

Adi 18218009

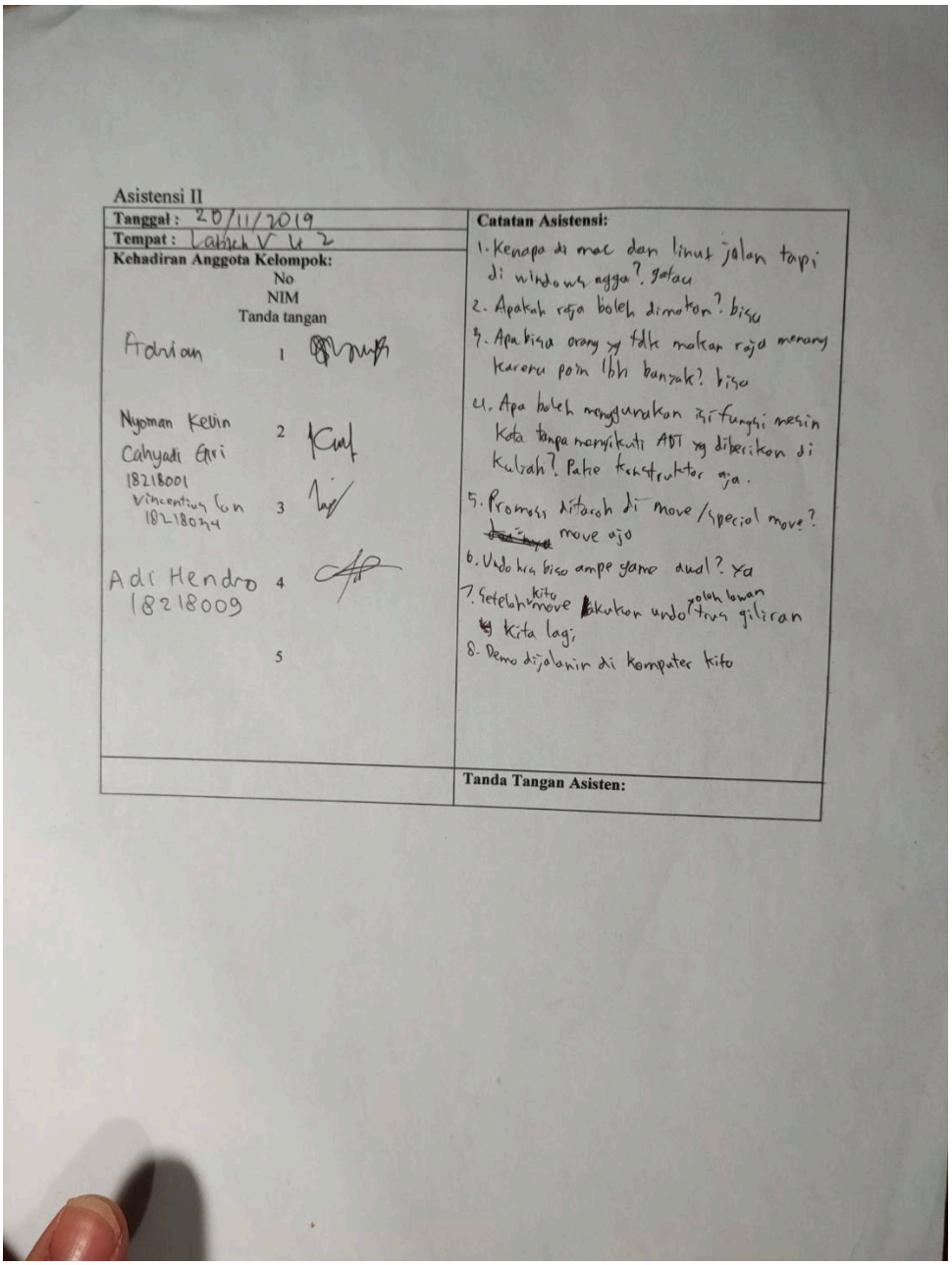
4 Ian 18218024

5

Catatan Asistensi:
Menu kata u/ menyimpan leaderboard. Mencatat permainan sebelumnya saja. Stack u/ undo nya point awal dan akhir. Fakemate sama telah memiliki manh ditanyakan. Untuk grafik pakai command line interface. File: driver main, .c, .h, .txt (u/ leaderboard), Bval.txt ada leaderboard, save. list linier simpan yang manh ada di papan. Kalo bidak dimakan dihapus.

Tanda Tangan Asisten:

Gambar 9.1 : Notulensi Asistensi 1



Asistensi II

Tanggal : 20/11/2019
Tempat : Lab Jar V lt 2
Kehadiran Anggota Kelompok:
No
NIM
Tanda tangan

1. Adrian
2. Nyoman Kevin Cahyadi Giri 18218001
3. Vincentius Gun 18218034
4. Adi Hendro 18218009
5.

Catatan Asistensi:

1. Kenapa di mac dan linux jalan tapi di windows nggak? gatau
2. Apakah raja boleh dimakan? bisa
3. Apa bisa orang yg tdk makan raja menang karena poin lbh banyak? bisa
4. Apa boleh menggunakan ai fungsi mesin kata tanpa mengikuti ADT yg diberikan di kuliah? Pake fungsi struktur aja.
5. Promosi ditaroh di move / special move? ~~bebas~~ move aja
6. Undo kra bisa ampe yang awal? ya
7. Setelah kita move dan lawan melakukan undo terus giliran kita lagi
8. Demo dijalanin di komputer kita

Tanda Tangan Asisten:

Gambar 9.2 : Notulensi Asistensi 2

9.3 Log Activity Anggota Kelompok

Tanggal	Kehadiran	Kegiatan
4/11/2019	-Kevin (18218001) -Adi (18218009) -Ian(18218034)	Membahas spesifikasi dan cara pengerjaan

9/11/2019	-Kevin (18218001) -Adrian (18218002) -Adi (18218009) -Ian(18218034)	Mulai pengerjaan. Membuat move.c, print papan, sebagian main
12/11/2019	-Kevin (18218001) -Adrian (18218002) -Adi (18218009) -Ian(18218034)	Mengerjakan special_move, undo, menyelesaikan line
18/11/2019	-Kevin (18218001) -Adrian (18218002) -Adi (18218009)	Menyelesaikan main dan ADT lain
25/11/2019	-Kevin (18218001) -Adrian (18218002) -Adi (18218009) -Ian(18218034)	Menyelesaikan bonus, debugging, membuat laporan

Tabel 9.1 : Log Act Anggota Kelompok