

LAPORAN TUGAS BESAR

IF2111 Algoritma dan Struktur Data

Aplikasi BNMO


Dipersiapkan oleh:

Kelompok 1

Wan Aufa Azis	18221001
Athira Dhyannis Taqir	18221022
Dhafin Ghalib Luqman Hakim	18221023
Syasya Umaira	18221026
Muhammad Mumtaz	18221029

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		<i>IF2111-TB2-03-01</i>		35
		<i>Revisi</i>	<i>1</i>	2 Desember 2022

Daftar Isi

1 Ringkasan	3
2 Penjelasan Tambahan Spesifikasi Tugas	4
3 Struktur Data (ADT)	4
3.1 ADT Array	4
3.2 ADT List	5
3.3 ADT Map	5
3.4 ADT Point	6
3.5 ADT Set	6
3.6 ADT Stack	6
3.7 ADT Matriks	7
4 Program Utama	7
5 Algoritma-Algoritma Menarik	7
6 Data Test	9
6.1 Data Test COMMAND DASAR	9
6.2 Data Test START	10
6.3 Data Test LOAD	10
6.4 Data Test SCOREBOARD	11
6.5 Data Test RESET SCOREBOARD	11
6.6 Data Test HISTORY	12
6.7 Data Test RESET HISTORY	13
6.8 Data Test PLAY GAME	13
■ 6.8.1 Data Test GAME HANGMAN	14
■ 6.8.2 Data Test GAME TOWER OF HANOI	15
■ 6.8.3 Data Test GAME SNAKE ON METEOR	16
7 Test Script	18
8 Pembagian Kerja dalam Kelompok	20
9 Lampiran	21
9.1 Deskripsi Tugas Besar 2	21
9.2 Notulen Rapat	31
9.3 Log Activity Anggota Kelompok	34

1 Ringkasan

Permasalahan yang dihadapi oleh Indra dan Doni adalah munculnya *bug* di sistem BNMO yang mereka perbaiki dua bulan yang lalu dan sekarang kami harus membantu mereka untuk memprogram ulang robot *video game console* kesayangan mereka. BNMO atau dibaca Binomo merupakan suatu aplikasi untuk menjalankan permainan. Aplikasi ini diprogram menggunakan bahasa C yang berbasis *command line interface*. Untuk membuat aplikasi ini, kami memanfaatkan berbagai struktur data (ADT). Fitur utama dari BNMO ini adalah memainkan *game*, menambahkan *game*, menghapus *game*, mengurutkan *game* yang akan dimainkan, menampilkan *game* yang telah dimainkan, dan menampilkan *scoreboard game*.

BNMO memiliki lima permainan utama yaitu RNG, Diner Dash, Hangman, Snake On Meteor, dan Tower of Hanoi. Permainan yang pertama adalah RNG atau *Random Number Generator* adalah permainan menebak angka acak yang sudah ditentukan oleh program. Saat permainan dimulai, program sudah menentukan angka acak yang harus ditebak oleh pemain. Selanjutnya, permainan yang kedua adalah Diner Dash. Diner Dash adalah permainan mengantar makanan sesuai dengan urutan prioritasnya. Pada permainan ini ada tiga *command* valid yaitu COOK untuk memasak makanan, SERVE untuk menyajikan makanan, dan SKIP untuk tidak melakukan apa apa namun terhitung telah menyelesaikan satu putaran permainan. Selanjutnya, permainan ketiga yaitu Hangman. Hangman merupakan permainan tebak kata dengan menyebutkan huruf dalam tebakan tertentu. Pemain akan memiliki kesempatan untuk menebak kata sebanyak sepuluh kali dan berkurang apabila pemain salah dalam menebak huruf.. Permainan akan selesai jika pemain kehabisan kesempatan untuk menebak kata. Selanjutnya, di permainan keempat ada game Tower of Hanoi. Tower of Hanoi merupakan *game* memindahkan piringan dengan posisi yang sama dengan peraturan piringan yang ada dibawah tidak boleh lebih kecil daripada piringan yang ada diatasnya. Di dalam permainan ini terdapat 3 tiang yaitu tiang a, b, dan c. Permainan dilakukan dengan memindahkan piringan dari tiang asal ke tiang tujuan sesuai yang diinginkan pemain. Selanjutnya, permainan yang kelima adalah Snake on Meteor yaitu permainan ular yang terinspirasi dari permainan Snake yang ada di sejak dulu di banyak konsol lama tetapi dengan tambahan adanya meteor yang menjatuhkan ular tersebut sebagai rintangan yang harus dihindari. Permainan akan berhenti jika komponen dari ular terkena meteor dan ekor baru ular tidak bisa dimunculkan lagi karena tidak terdapat area yang cukup..

BNMO juga kami program untuk memiliki permainan tambahan yaitu Jari Bocil. Permainan yang terinspirasi dari permainan tradisional dengan menggunakan 10 jari yang dimodifikasi. Dalam permainan ini, pemain akan berhadapan dengan komputer untuk membuat jumlah jari di kedua tangan lawan menjadi sama sama 5. Permainan akan berakhir jika jari di kedua tangan pemain atau komputer berjumlah sama sama 5.

Pengerjaan tugas besar ini membantu kami untuk memahami lebih lagi tentang bahasa C dan mengaplikasikan materi materi dasar yang diajarkan di kelas. Tugas besar ini bermanfaat bagi kami karena membuat kami belajar untuk menemukan ide dan alur dari membuat program hingga selesai.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 Spesifikasi Fitur Tambahan 1

Kami membuat fitur tambahan dalam permainan Hangman yaitu adanya penambahan kamus. Kami menambahkan 2 menu saat *user* memainkan permainan yaitu pilihan untuk bermain langsung dan pilihan untuk menambah sejumlah kata ke dalam kamus yang berisi daftar kata dari permainan Hangman yang nantinya bisa dibaca oleh program. Jika kata belum terdapat di dalam kamus maka kata akan ditambahkan ke dalam kamus sementara jika kata sudah terdapat dalam kamus maka program akan menyuruh *user* untuk memasukkan kata lain.

2.2 Spesifikasi Fitur Tambahan 2

Kami membuat fitur tambahan dalam permainan Tower of Hanoi yaitu membuat program untuk meminta *input*-an opsi jumlah piringan yang digunakan. Nantinya, skor akan menyesuaikan dengan jumlah piringan mulainya. Pada program kami, tidak ada jumlah maksimum untuk jumlah piringan yang bisa di-*input* oleh *user*.

2.3 Spesifikasi Fitur Tambahan 3

Kami membuat fitur tambahan dalam permainan Snake on Meteor yaitu membuat sisi peta yang berseberangan berhubungan sehingga ular dalam permainan ini bisa menembus keluar dari *maps* dan muncul dari sisi seberang. Selain itu kami juga menambahkan *obstacle* yang tidak bisa ditembus oleh ular dan *obstacle* ini tidak akan berpindah tempat selama permainan berlangsung.

3 Struktur Data (ADT)

Pada pembuatan aplikasi BNMO ini, kami menggunakan beberapa struktur data untuk menyelesaikan persoalan-persoalan pada Tugas Besar yaitu ADT Arraymap, ADT List, ADT Map, ADT Matriks, ADT Point, ADT Set, ADT Stack.

3.1 ADT Array

Pada ADT Array tubes 2 ini kami menambahkan 2 file tambahan yang terdiri dari satu *file header* yaitu arraymap.h dan satu file implementasi dari di *file header* berupa *file c* yaitu arraymap.c. Dalam ADT Array strukturnya terdiri dari sebuah *pointer A* yang menyimpan nilai bertipe *map*, *IdxType* bertipe *integer*, *Capacity* bertipe *integer*, *Neff* menyimpan nilai efektif elemen bertipe *integer*. ADT arraymap digunakan untuk menyimpan *scoreboard* dari tiap *game* yang berbentuk *array of map*. ADT arraymap memiliki 12 fungsi primitif, yaitu *Makearraymap()* untuk membuat *arraymap* kosong dengan ukuran *initialsize*, *Deallocatearraymap()* untuk mendealokasi *A* dalam *arraymap*, *IsEmptyarraymap()* fungsi untuk mengetahui apakah suatu *arraymap* kosong, *Lengtharraymap()* fungsi untuk mendapatkan banyaknya elemen efektif *arraymap*, 0 jika tabel kosong, *Getmap()* untuk mengembalikan elemen *arraymap* *L* yang ke-*I*, *GetCapacityarraymap()* fungsi untuk mendapatkan kapasitas yang tersedia, *InsertAtarraymap()* fungsi untuk menambahkan elemen baru di akhir *arraymap*,

InsertLastarrmap() fungsi untuk menambahkan elemen baru di akhir arraymap, InsertFirstarrmap() untuk menambahkan elemen baru di awal arraymap, DeleteAtarrmap() fungsi untuk menghapus elemen di index ke-i arraymap, DeleteLastarrmap() fungsi untuk menghapus elemen terakhir arraymap, DeleteFirstarrmap() fungsi untuk menghapus elemen pertama arraymap, Printarraymap() fungsi untuk melakukan print suatu arraymap.

3.2 ADT List

Dalam ADT List strukturnya terdiri dari sebuah *pointer address*, *infotype* pada info, *address* untuk *next*, *address* untuk *prev*, *address* untuk *first*, dan *address* untuk *last*. ADT List ini memiliki 20 fungsi primitif, yaitu IsEmptylist() mengirim *true* jika list kosong, CreateEmptylist() untuk membuat *list* kosong, Alokasi() untuk mengirim *address* hasil alokasi sebuah elemen, Dealokasi() untuk melakukan pengembalian *address*, Searchlist() untuk mencari apakah ada elemen list dengan info(P) = X, InsVFirst() untuk menambahkan elemen pertama dengan nilai X jika alokasi berhasil, InsVLast() untuk menambahkan elemen *list* di akhir dengan elemen baru, DelVFirst() untuk menghapus elemen pertama, nilai info akan disimpan pada X dan alamat elemen pertama di-dealokasi, DelVLast() untuk menghapus elemen terakhir, nilai info akan disimpan pada X dan alamat elemen terakhir di-dealokasi, InsertFirstlist() untuk menambahkan elemen ber-address P sebagai elemen pertama, InsertLastlist() untuk menambahkan P sebagai elemen terakhir yang baru, InsertAfter() untuk *insert* P sebagai elemen sesudah elemen beralamat Prec, InsertBefore() untuk *insert* P sebagai elemen sebelum elemen beralamat Succ, DelFirstlist() untuk mengubah *first* elemen yang baru dengan suksesor elemen pertama yang lama, DelLastlist() untuk mengubah *last* elemen yang baru dengan suksesor elemen pertama yang lama (jika ada), Delp() untuk menghapus P dari *list* dan didealokasi, DelAfter() untuk menghapus alamat elemen *list* Next(Prec), DelBefore() untuk menghapus alamat elemen Prev(Succ), PrintForward() untuk mencetak isi *list* dari elemen pertama, PrintBackward() untuk mencetak isi *list* dari elemen terakhir. ADT List berguna pada *game* Snake on Meteor karena bisa menyimpan tipe *point* yang tidak berurut pada badan ular. ADT List diimplementasikan sebagai ADT List dengan nama *file header* “listdp.h”.

3.3 ADT Map

Dalam ADT Map strukturnya terdiri dari *key* yang bertipe *word*, *value* yang bertipe *integer*, dan *count* yang bertipe *integer*. ADT Map memiliki 8 fungsi primitif yaitu CreateEmptymap() untuk membuat sebuah Map M kosong berkapasitas MaxEl, IsEmptymap() untuk mengirim *true* jika Map M kosong, IsFullmap() untuk mengirim *true* jika Map M penuh, Value() untuk mengembalikan nilai *value* dengan *key* K dari M, Insertmap() untuk menambahkan elemen sebagai elemen Map M dengan v yang berurutan dari terbesar, Deletemap() untuk menghapus elemen dari Map M, IsMembermap() untuk mengembalikan *true* jika k adalah member dari M, dan CetakMap() untuk mencetak Map M ke layar. ADT Map untuk membantu menyimpan *scoreboard* pada *key* kita dapat menyimpan nama *player* yang bersifat unik dan *value* untuk menyimpan *score* dari setiap *player*. ADT Map diimplementasikan sebagai ADT Map dengan nama *file header* “map.h”.

3.4 ADT Point

Dalam ADT Point strukturnya terdiri dari X (absis) yang bertipe *integer* dan Y (ordinat) yang bertipe *integer*. Pada ADT Point memiliki 12 fungsi primitif, yaitu MakePOINT() untuk membentuk sebuah POINT dari komponen-komponen, TulisPOINT() untuk menulis ke layar dengan format “(X,Y)”, ComparePOINT() untuk mengirim *true* jika kedua point berada pada absis dan ordinat yang sama, EQ() untuk mengirimkan *true* jika P1=P2 memiliki absis dan ordinatnya sama, NEQ() untuk mengirim *true* jika P1 tidak sama dengan P2, IsOrigin() untuk menghasilkan *true* jika P adalah titik origin, IsOnSbX() untuk menghasilkan *true* jika P terletak pada sumbu X, IsOnSbY() untuk menghasilkan *true* jika P terletak pada sumbu Y, Kuadran() untuk menghasilkan kuadran dari P, PlusDelta() untuk mengirim salinan P yang absisnya adalah Absis(P) + deltaX dan ordinatnya adalah Ordinat(P) + deltaY, Geser() untuk menggeser absis sebesar deltaX dan ordinatnya sebesar deltaY, ReplacePOINT() untuk mengganti absis dan ordinat *point* P dengan X dan Y. Alasan digunakannya ADT Point ini karena dapat membantu merepresentasikan indeks dari matriks pada *game* Snake on Meteor yang nantinya akan digunakan untuk mencetak peta permainan tersebut. ADT Point diimplementasikan sebagai ADT Point dengan *file header* “point.h”.

3.5 ADT Set

Dalam ADT Set strukturnya terdiri dari *TypeSet* dengan elementnya 26 yang bertipe *char* dan *idx* dengan *count* yang bertipe *integer*. Pada ADT Set memiliki 7 fungsi primitif yaitu CreateEmptyset() untuk membuat *Set* S kosong berkapasitas *MaxEl*, IsEmptyset() untuk mengirim *true* jika *Set* S kosong, IsFullset() untuk mengirim *true* jika *Set* S penuh, Insertset() untuk menambahkan elemen sebagai elemen *Set* S, Deletset() untuk menghapus elemen dari *Set* S, IsMemberset() untuk mengembalikan *true* jika elemen adalah member dari *Set* S, PrintSet() untuk mencetak isi *Set* S ke layar. Alasan digunakan ADT Set ini karena pada *game* Hangman untuk menyimpan karakter-karakter yang sudah ditebak oleh *player*. ADT Set diimplementasikan sebagai ADT Set dengan *file header* “set.h”.

3.6 ADT Stack

Dalam ADT Stack strukturnya terdiri dari *TypeStack* dengan tabel yang menyimpan 100 elemen bertipe *word* dan TOP yang bertipe *integer*. Pada ADT Stack memiliki 8 fungsi primitif yaitu CreateEmptystack() untuk membuat sebuah *stack* S yang kosong berkapasitas *MaxEl*, IsEmptystack() untuk mengirim *true* jika *Stack* kosong, IsFullstack() untuk mengirim *true* jika tabel penampung nilai elemen *stack* penuh, Push() untuk menambahkan X sebagai elemen *Stack* S, Pop() untuk menghapus X dari *Stack* S, Reversestack() untuk menghasilkan *Stack* yang merupakan kebalikan dari *Stack input* S, CetakStack() untuk mencetak *Stack* S ke layar, dan DeleteHistory() untuk menghapus catatan *history game* yang ingin di hapus. Alasan digunakan ADT Stack ini karena dapat membantu dalam menampilkan *history game*. ADT Stack diimplementasikan sebagai ADT Stack dengan *file header* “stack.h”.

Dalam ADT Stacktoh strukturnya terdiri dari *T untuk menyimpan elemen bertipe *integer*, SCapacity bertipe *integer*, dan TOP bertipe *integer*. Pada ADT Stacktoh memiliki 6 fungsi primitif yaitu CreateEmptystacktoh(), IsEmptystacktoh(), IsFullstacktoh(), Pushtoh(), Poptoh(), dan Displaystacktoh(). Alasan digunakan ADT Stacktoh ini karena dapat membantu

dalam mengimplementasikan *game* Tower of Hanoi dengan tipe data *integer* yang menggambarkan jumlah piringan. ADT Stacktoh diimplementasikan sebagai ADT Stacktoh dengan *file header* “stacktoh.h”.

3.7 ADT Matriks

Dalam ADT Matriks strukturnya terdiri dari *array* yang bertipe *char* dan menyimpan 5 ruang. Pada ADT Matriks memiliki 7 fungsi primitif yaitu MakeMatriks() untuk membuat matriks kosong dengan ukuran *initialsise*, IsEmptyMatriks() untuk mengetahui apakah suatu matriks kosong, IsFullMatriks() untuk mengecek apakah sebuah matriks sudah penuh atau belum, GetelmtMatriks() untuk mengembalikan elemen matriks L yang ke-I, InsertAtMatriks() untuk menambahkan elemen di index ke-I matriks, DeleteAtMatriks() untuk menghapus elemen di index ke-i matriks, PrintMatriks() untuk mencetak suatu matriks. Alasan digunakan ADT Matriks ini karena untuk mencetak peta pada *game* Snake on Meteor yaitu berupa *array of array*, yang menyimpan kondisi peta yang sesungguhnya (representasi dari peta permainan). ADT Matriks diimplementasi sebagai ADT Matriks dengan *file header* “matriks.h”.

4 Program Utama

Program utama dengan file “main.c” meng-include semua file ADT yang ada. Program utama dari aplikasi BNMO dimulai dengan menampilkan *main menu* yang berisikan *welcome page* dan menampilkan pilihan *menu* untuk *start* dan *load game*. Dengan Prosedur CHOOSEMODE *user* bisa memilih untuk melakukan *start* atau *load game*. Setelah itu, *main menu* akan menerima *input* dari *user* untuk lanjut memainkan permainan baru atau melanjutkan permainan yang sudah pernah dimainkan sebelumnya. Apabila *user* memilih *start* maka program akan melanjutkan dengan mengakses *file config*, namun jika *user* memilih *load* maka program akan melanjutkan dengan mengakses *file game user* sebelumnya. Selanjutnya program akan menampilkan fitur-fitur yang ada yaitu CREATEGAME, LISTGAME, DELETEGAME, QUEUEGAME, PLAYGAME, SKIPGAME, HISTORY, RESET HISTORY, SCOREBOARD, RESET SCOREBOARD, SAVE, HELP, dan QUIT lalu program akan terus berjalan sampai *user* memilih fitur QUIT. Pada tugas besar 2 ini fitur yang bertambah adalah HISTORY, RESET HISTORY, SCOREBOARD, dan RESET SCOREBOARD. Terdapat tambahan untuk *game* juga yaitu Hangman, Snake on Meteor, dan Tower of Hanoi.

5 Algoritma-Algoritma Menarik

Di dalam program yang kami buat, terdapat beberapa algoritma yang bisa dikategorikan sebagai algoritma yang menarik.

5.1 Algoritma 1

Algoritma yang kami maksud adalah prosedur EndGame. Prosedur ini kami anggap menarik karena algoritma ini ada sebagai penunjang dari prosedur PLAYGAME yang akan

menunjukkan *scoreboard* dari semua *game* yang dimainkan dan meminta *input* nama pemain yang jika nama pemain sebelumnya sudah di-*input* akan meminta *input*-an nama yang lain sedangkan jika nama pemain belum ada akan ditambahkan kedalam *scoreboard*.

Prosedur EndGame

```
void EndGame(arraymap *ScoreBoardGame, int Game, int score) {
    printf("SKOR AKHIR : %i\n", score);
    label : printf("MASUKKAN NAMA: "); STARTWORD(); printf("\n");
    boolean found = false; int j = 0;
    while (!found && j<ScoreBoardGame->A[Game].Count) {
        if (WordCompareKapital(currentWord, ScoreBoardGame->A[Game].Elements[j].Key)) {
            found = true; /*kondisi kalau ketemu yang sama namanya*/
        }
        else {
            j++; /*kondisi kalau nama nya gaada yang sama*/
        }
    }
    if (found == true) { /*ketemu nama yang sama*/
        printf("Nama yang di Input sudah ada, silahkan masukkan nama lain!\n");
        goto label;
    }
    else { /*Nama yang di input tidak ada di ScoreBoard*/
        Insertmap(&(*ScoreBoardGame).A[Game], currentWord, score);
    }
}
```

5.2 Algoritma 2

Algoritma yang menarik yang ada dalam program kami adalah ADT Arraymap dan ADT Matriks. Penggunaan ADT *arraymap* ini untuk *array* yang menyimpan *maps scoreboard* dari setiap gamenya dan penggunaan ADT Matriks pada permainan Snake on Meteor yang berfungsi untuk merepresentasikan peta sesungguhnya pada permainan.

5.3 Algoritma 3

Algoritma ketiga yang menurut kami menarik adalah fungsi *Score_Counter* pada permainan Tower of Hanoi. Fungsi ini berguna untuk menghitung *maximum moves* yang bisa dilakukan oleh *user* pada permainan ini.

Fungsi Score_Counter

```
int Score_Counter(int counter, int JumlahPiringan) {
    int score, minscore = 0, minmoves = 1;
```



```

for (int i = 0; i<JumlahPiringan;i++){
    minmoves = minmoves*2;
} /*looping untuk pembagi counter nantinya*/

    int maxscore = JumlahPiringan*2; /*maxscore akan bertambah
dan berkurang bergantung Jumlah Piringan*/

    intmaxmoves=(minmoves-1)*((JumlahPiringan*2)+1);/*perhitung
an maxmoves, jika lebih score akan nol*/

    if (counter > maxmoves) {return minscore;} /*kasus pemain
movesnya melebihi maxmoves*/
    else { /*kalau movesnya masih lebih kecil dari maxmoves*/
        score = maxscore-((counter/(minmoves-1))-1);
        return score;
    }
}

```

6 Data Test

6.1 Data Test COMMAND DASAR

Pada tes ini dilakukan pengujian untuk memastikan bahwa *command* dasar program BNMO dapat berjalan. Pengujian *command* dasar telah dilakukan pada saat laporan tugas besar 1 IF2111 Algoritma dan Struktur Data. Berikut salah satu contoh *test command* dasar berupa tampilan judul dan *main menu*.



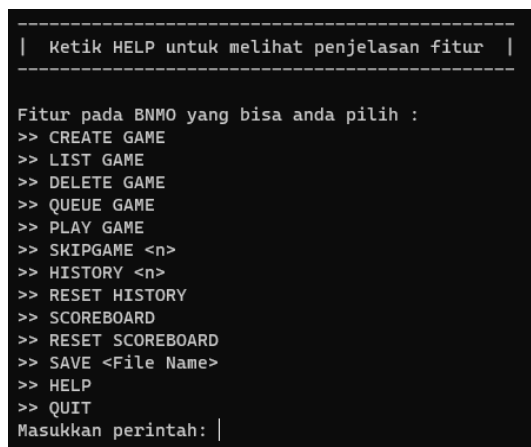
Gambar 1. Tampilan Judul dan Main Menu

6.2 Data Test START

Pada tes ini akan menjalankan *input start* serta memastikan fungsi tersebut dapat berjalan. Adapun pada gambar 2 merupakan tampilan saat akan memasukkan *input* dan gambar 3 merupakan hasil tampilan setelah memasukkan *input*.



Gambar 2. Tampilan Saat Memasukkan *Input* START



Gambar 3. Tampilan Setelah Memasukkan *Input* START

6.3 Data Test LOAD

Pada tes ini akan menjalankan *input load* serta memastikan fungsi tersebut dapat berjalan. Adapun pada gambar 4 merupakan tampilan saat akan memasukkan *input* dan hasil tampilan dari *input*-an tersebut akan sama dengan gambar 3.

```

-----| MAIN MENU |-----
[1] START
[2] LOAD <File Name>
Silahkan memilih mode START/LOAD <File Name>: LOAD SAVE2.TXT|

```

Gambar 4. Tampilan Saat Memasukkan Load File save2.txt

6.4 Data Test SCOREBOARD

Pada tes ini akan dilakukan pengecekan *scoreboard* dari permainan yang pernah dimainkan oleh para pengguna di aplikasi BNMO. Setiap pengguna yang mengalami keadaan memenangkan *game* atau permainan tersebut telah berakhir maka pengguna akan diminta untuk meminta memasukkan nama yang tidak terdapat pada *scoreboard* yang telah ada. Untuk melihat keadaan *scoreboard*, pengguna dapat memasukkan *command* SCOREBOARD dan akan muncul tampilan seluruh *scoreboard* permainan yang ada seperti pada gambar 5.

```

Masukkan perintah: SCOREBOARD

**** SCOREBOARD GAME RNG ****
| Nama      | Score |
|-----|
| Azis      | 100   |
| Muntaz    | 90    |
|-----|

**** SCOREBOARD GAME Diner DASH ****
| Nama      | Score |
|-----|
| aku       | 25000 |
| saya      | 17000 |
| kita      | 15500 |
| dia       | 9000  |
| mereka    | 0     |
|-----|

**** SCOREBOARD GAME HANGMAN ****
| Nama      | Score |
|-----|
| syasya    | 98    |
| ghalib    | 20    |
| Azis      | 20    |
| thira     | 13    |
|-----|

**** SCOREBOARD GAME TOWER OF HANOI ****
| Nama      | Score |
|-----|
| ekhem     | 22    |
|-----|

**** SCOREBOARD GAME SNAKE ON METEOR ****
| Nama      | Score |

```

Gambar 5. Tampilan Scoreboard

6.5 Data Test RESET SCOREBOARD

Pada tes ini akan mereset scoreboard dari permainan di aplikasi BNMO. Untuk mereset *scoreboard*, pengguna akan memasukkan *command* berupa RESET SCOREBOARD. Setelah memasukkan *command* tersebut, akan muncul tampilan seperti gambar 6 dan pengguna akan

memilih nomor *scoreboard* yang akan dihapus. Setelah berhasil *reset scoreboard* akan muncul tampilan seperti gambar 7. Namun, saat pengguna tidak jadi untuk *reset scoreboard* akan menampilkan gambar 8.

```
Masukkan perintah: RESET SCOREBOARD

DAFTAR SCOREBOARD :
0. ALL
1. RNG
2. Diner DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. Jari Bocil
7. Custom 1
8. Custom 2

MASUKKAN NOMOR SCOREBOARD YANG INGIN DIHAPUS: |
```

Gambar 6. Tampilan Saat Reset Scoreboard

```
MASUKKAN NOMOR SCOREBOARD YANG INGIN DIHAPUS: 0

APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD ALL? (Y/N) Y

Scoreboard berhasil di-reset.
Tekan <ENTER> untuk melanjutkan >>> |
```

Gambar 7. Tampilan Ketika Berhasil Reset Scoreboard

```
MASUKKAN NOMOR SCOREBOARD YANG INGIN DIHAPUS: 0

APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD ALL? (Y/N) N

Scoreboard gagal di-reset.
Tekan <ENTER> untuk melanjutkan >>> |
```

Gambar 8. Tampilan Ketika Tidak Jadi Reset Scoreboard

6.6 Data Test HISTORY

Pada tes ini akan melihat permainan apa saja yang telah dimainkan oleh pengguna berdasarkan urutan teratas *game* yang terakhir dimainkan. Untuk menjalankan *history*, pengguna akan memasukkan *command* HISTORY <n> dengan <n> merupakan jumlah permainan yang telah dimainkan. Setelah memasukkan *command* tersebut akan muncul tampilan seperti gambar 9.

```

Masukkan perintah: HISTORY 5

Berikut adalah daftar Game yang telah dimainkan
1. Diner DASH
2. TOWER OF HANOI
3. Jari Bocil
4. Custom 1
5. Diner DASH
Tekan <ENTER> untuk melanjutkan >>> |

```

Gambar 9. Tampilan History

6.7 Data Test RESET HISTORY

Pada tes ini akan menghapus semua *history* permainan pernah dilakukan oleh pengguna. Untuk menjalankan *reset history*, pengguna akan memasukkan *command* RESET HISTORY dan jika berhasil maka seluruh data *history* akan terhapus seperti pada gambar 10.

```

Masukkan perintah: RESET HISTORY

APAKAH KAMU INGIN MELAKUKAN RESET HISTORY? (Y/N)
Y
History berhasil di-reset.
Tekan <ENTER> untuk melanjutkan >>> |

```

Gambar 10. Tampilan Reset History

6.8 Data Test PLAY GAME

Pada menjalankan fitur *play game*, pengguna harus memasukkan *game* terlebih dahulu dengan fitur *queue game*. Jika belum terdapat *game* di antrian maka akan muncul tampilan seperti gambar 11. Akan tetapi jika terdapat antrian *game* pada fitur *play game* maka akan muncul tampilan sesuai gambar 12.

```

Berikut adalah daftar antrian game-mu:

Tidak ada game di dalam antrian untuk dimainkan, silahkan masukkan game kedalam antrian terlebih dahulu!

```

Gambar 11. Tampilan Saat Tidak Terdapat *Game* Untuk Dijalankan

```

Masukkan perintah: PLAY GAME

Berikut adalah daftar antrian game-mu:
1. RNG
2. Diner DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. Jari Bocil

Loading...

```

Gambar 12. Tampilan Jika Terdapat Antrian *Game*

■ 6.8.1 Data Test GAME HANGMAN

Pada tes ini akan menjalankan *game Hangman*. Saat pengguna menjalankan *game* ini akan muncul tampilan awal seperti gambar 13 yang dimana pengguna akan memilih tiga opsi tersebut. Pilihan 0 untuk keluar dari permainan, pilihan 1 untuk memasukkan tebakan baru, dan pilihan 2 untuk memainkan *game Hangman*. Untuk tampilan pilihan 1 akan seperti gambar 14 dan tampilan pilihan 2 akan seperti gambar 15. Adapun saat pengguna kalah maka tampilan seperti gambar 16 dan saat pengguna menamatkan permainan dan memilih opsi 0 akan muncul tampilan gambar 17.

```
Mode game yang tersedia:
0.Exit
1.In-game dictionary
2.Play Hangman
Silahkan masukkan mode : |
```

Gambar 13. Tampilan Awal Permainan Hangman

```
Mode game yang tersedia:
0.Exit
1.In-game dictionary
2.Play Hangman
Silahkan masukkan mode : 1
Silahkan input nama kota baru untuk dimasukkan kedalam dictionary (PASTIKAN SEMUA HURUF ADALAH KAPITAL) : LOMBOK
Kata berhasil dimasukkan kedalam dictionary!
```

Gambar 14. Tampilan Saat Pilihan 1

```
=====SELAMAT DATANG DI GAME HANGMAN=====
      UJI KEAHLIAN-MU DENGAN MENEBAK KATA!
      SELAMAT BERMAIN DAN SEMOGA BERUNTUNG!! :)

-----

HINT : KATA MERUPAKAN KOTA TERKENAL DI INDONESIA!
Tebakan sebelumnya :-
Kata :-----
Kesempatan : 10
Masukkan tebakan : |
```

Gambar 15. Tampilan Saat Pilihan 2

```
HINT : KATA MERUPAKAN KOTA TERKENAL DI INDONESIA!
Tebakan sebelumnya :-
Kata :-----
Kesempatan : 1
Masukkan tebakan : o
GameOver! Kesempatan-mu sudah habis!

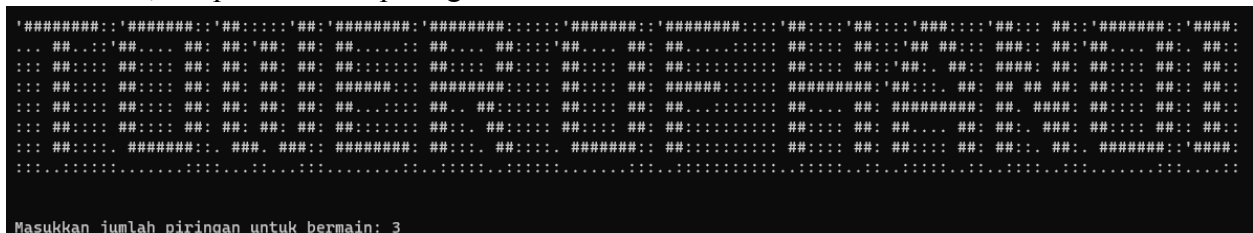
Mode game yang tersedia:
0.Exit
1.In-game dictionary
2.Play Hangman
Silahkan masukkan mode : |
```

```
HINT : KATA MERUPAKAN KOTA TERKENAL DI INDONESIA!
Tebakan sebelumnya :AOSL
Kata :SOLO_
Kesempatan : 3
Masukkan tebakan : k

Selamat kamu berhasil menebak SOLOK! Kamu mendapatkan 5 point!
GameOver! Selamat, kamu telah menghabiskan seluruh dictionary!

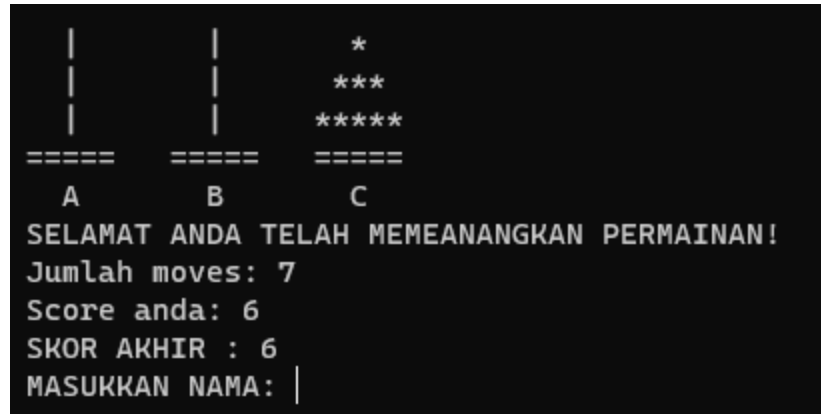
Mode game yang tersedia:
0.Exit
1.In-game dictionary
2.Play Hangman
Silahkan masukkan mode : 0
SKOR AKHIR : 249
MASUKKAN NAMA: Muntaz
```

Pada tes ini akan menjalankan *game* Tower of Hanoi. Saat pengguna menjalankan *game* ini akan muncul tampilan awal seperti gambar 18 yang dimana pengguna akan jumlah kepingan yang ingin diselesaikan. Setelah memasukkan jumlah kepingan, pemain akan ditampilkan seperti gambar 19 untuk memainkan permainan. Jika permainan telah usai, tampilan akan seperti gambar 20.



```
Masukkan jumlah piringan untuk bermain: 3
  *      |      |
 ***     |      |
*****   |      |
=====  |      |
  A      B      C
TIANG ASAL: |
```

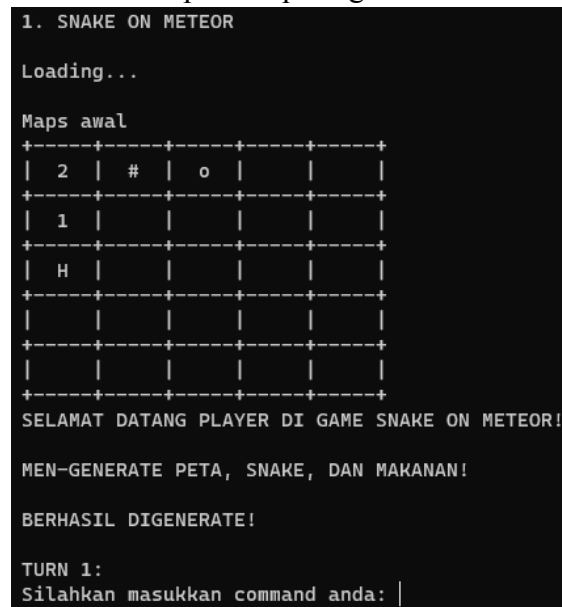
STEI- ITB	IF2111-TB2-03-01	Halaman 15 dari 34 halaman
<p>Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB</p>		



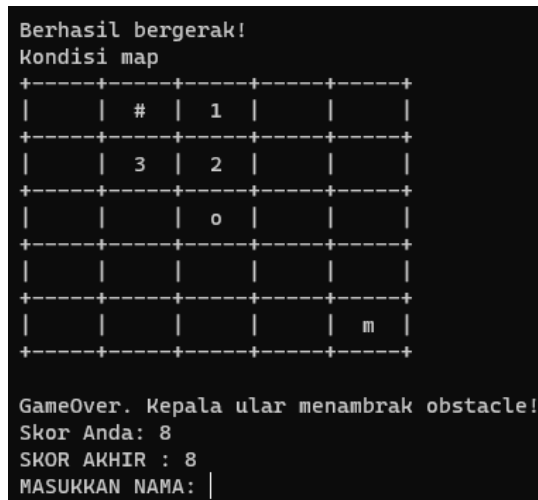
Gambar 20. Tampilan Saat Permainan Selesai

■ 6.8.3 *Data Test* **GAME SNAKE ON METEOR**

Pada tes ini akan menjalankan *game* Snake on Meteor. Saat pengguna menjalankan *game* ini akan muncul tampilan awal seperti gambar 21. Adapun saat pemain kalah akan memiliki tampilan seperti gambar 22.



Gambar 21. Tampilan Saat *Game* Dimulai



Gambar 22. Tampilan Saat Pemain Kalah

7 Test Script

No .	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1.	Fitur COMMAND DASAR	Memeriksa dan memastikan apakah command dasar program BNMO bisa berjalan.	Pada saat menjalankan program, akan keluar <i>main menu</i>	Data Test 6.1	Memunculkan <i>main menu</i> yang terdapat 2 pilihan yaitu <i>START</i> atau <i>LOAD GAME</i>	Sesuai yang diharapkan
2	Fitur START	Memeriksa apakah fitur <i>START</i> bisa berjalan dengan baik dan permainan bisa dimulai dan dijalankan	Memasukkan <i>command START</i> pada <i>main menu</i>	Data Test 6.2	Memunculkan fitur pada BNMO yang dapat dipilih, kemudian akan diminta <i>input</i> fitur	Sesuai yang diharapkan
3	Fitur LOAD	Memeriksa apakah permainan bisa di <i>load</i> atau fitur <i>LOAD</i> bisa	Memasukkan <i>command Load</i> pada main menu untuk <i>file</i> yang sudah di <i>save</i>	Data Test 6.3	Berhasil mengakses <i>file</i> yang telah disimpan oleh	Sesuai yang diharapkan

		berhasil dijalankan				
4	Fitur SCOREBOARD	Memeriksa apakah fitur <i>SCOREBOARD</i> bisa berjalan dengan baik	Memasukkan <i>command SCOREBOARD</i> dan meng- <i>input</i> nama pemain	Data Test 6.4	Memunculkan <i>scoreboard</i> yang ada dan sesuai spesifikasi	Sesuai yang diharapkan
5.	Fitur RESET SCOREBOARD	Memeriksa apakah scoreboard bisa di reset dengan fitur <i>RESET SCOREBOARD</i>	Memasukkan <i>command RESET SCOREBOARD</i>	Data Test 6.5	Berhasil melakukan reset dan memilih <i>scoreboard</i> yang ingin dihapus	Sesuai yang diharapkan
6.	Fitur HISTORY	Memeriksa apakah fitur <i>HISTORY</i> bisa dijalankan	Memasukkan <i>command HISTORY</i>	Data Test 6.6	Menampilkan permainan apa saja yang telah dimainkan berdasarkan urutan teratas <i>game</i> yang terakhir dimainkan	Sesuai yang diharapkan
7.	Fitur RESET HISTORY	Memeriksa apakah fitur <i>RESET HISTORY</i> bisa dijalankan	Memasukkan <i>command RESET HISTORY</i>	Data Test 6.7	Menghapus semua <i>history</i> permainan yang pernah dilakukan oleh pengguna	Sesuai yang diharapkan
8.	Fitur Game HANGMAN	Memeriksa apakah <i>game</i> Hangman bisa berjalan dengan baik	Memasukkan <i>game</i> HANGMAN ke dalam <i>QUEUE GAME</i> , lalu <i>input PLAY GAME</i> untuk memainkan <i>game</i>	Data Test 6.8.1.	<i>Game</i> Hangman berhasil dimainkan sesuai dengan ketentuan <i>game</i>	Sesuai yang diharapkan
7.	Fitur Game TOWER OF HANOI	Memeriksa apakah <i>game</i> Tower Of Hanoi bisa berjalan dengan baik	Memasukkan <i>game</i> TOWER OF HANOI ke dalam <i>QUEUE GAME</i> , lalu <i>input PLAY GAME</i> untuk memainkan <i>game</i>	Data Test 6.8.2	<i>Game</i> Tower of Hanoi berhasil dimainkan sesuai dengan ketentuan <i>game</i>	Sesuai yang diharapkan

8.	Fitur Game SNAKE ON METEOR	Memeriksa apakah <i>game</i> Snake On Meteor bisa berjalan dengan baik	Memasukkan <i>game</i> SNAKE ON METEOR ke dalam <i>QUEUE GAME</i> , lalu <i>input PLAY GAME</i> untuk memainkan <i>game</i>	Data Test 6.8.3	<i>Game</i> Snake On Meteor berhasil dimainkan sesuai dengan ketentuan <i>game</i>	Sesuai yang diharapkan
----	----------------------------	--	---	-----------------	--	------------------------

8 Pembagian Kerja dalam Kelompok

No.	Fitur/ADT	NIM Coder	NIM Tester
1	ADT List	18221001	18221001, 18221022, 18221023, 18221026, 18221029
2	ADT Map	18221001	18221001, 18221022, 18221023, 18221026, 18221029
3	ADT Matriks	18221001	18221001, 18221022, 18221023, 18221026, 18221029
4	ADT Point	18221001	18221001, 18221022, 18221023, 18221026, 18221029
5	ADT Set	18221001	18221001, 18221022, 18221023, 18221026, 18221029
6	ADT Stack	18221001, 18221023	18221001, 18221022, 18221023, 18221026, 18221029
7	Scoreboard & Reset Scoreboard	18221029	18221001, 18221022, 18221023, 18221026, 18221029
8	History & Reset History	18221026	18221001, 18221022, 18221023, 18221026, 18221029
9	Game Hangman	18221022	18221001, 18221022, 18221023, 18221026, 18221029
10	Game Tower of Hanoi	18221023	18221001, 18221022, 18221023, 18221026, 18221029

11	Game Snake on Meteor	18221001	18221001, 18221022, 18221023, 18221026, 18221029
----	----------------------	----------	--

9 Lampiran

9.1 Deskripsi Tugas Besar 2

Command


Terdapat beberapa aturan umum command yang digunakan:

1. Semua command yang valid harus berupa **huruf kapital**. Mahasiswa dibebaskan apabila ingin melakukan handling terhadap huruf kecil.
2. Command yang tidak sesuai dengan ketentuan yang disebutkan pada bagian dibawah dianggap tidak valid. Handling command tidak valid dibebaskan kepada mahasiswa (boleh meminta ulang command yang sama atau meminta command baru)

Pada setiap giliran, pemain dapat memasukkan command-command berikut:

a. Command Dasar

Semua command yang terdapat pada

 Spesifikasi Tugas Besar 1 IF2111 2022/2023

b. SCOREBOARD

- Di setiap keadaan *game over* atau menang sebuah game, program meminta nama pemain.
- Nama pemain yang valid adalah **nama yang belum terpakai di scoreboard game yang sedang dimainkan**. Kemudian program menyimpan nama dan skor *game* tersebut di ADT Map.
- Perintah SCOREBOARD merupakan command yang digunakan untuk melihat nama dan skor untuk semua game.
- Urutan *scoreboard game* yang ditampilkan mengikuti **urutan pada command LIST GAME**.
- Urutan nama pada *scoreboard* diurutkan berdasarkan skor. Skor tertinggi berada di urutan pertama dan yang terendah berada

di urutan terakhir. Jika ada skor yang sama, skor yang lebih dulu dimasukkan ke *scoreboard* ditampilkan duluan.

c. **RESET SCOREBOARD**

RESET SCOREBOARD merupakan command yang digunakan untuk melakukan reset terhadap scoreboard permainan. Reset dapat dilakukan untuk menghapus semua informasi pada setiap permainan maupun memilih salah satu permainan untuk di-reset.

d. **HISTORY <n>**

HISTORY <n> merupakan command yang digunakan untuk melihat permainan apa saja yang telah dimainkan dari data yang sudah ada dari file konfigurasi (**Jika LOAD**) dan dari mulai Start Game juga, dengan <n> adalah jumlah permainan yang telah dimainkan yang ingin ditampilkan. Urutan teratas merupakan permainan terakhir yang dimainkan. Jika <n> lebih besar dari jumlah permainan yang telah dimainkan, akan menampilkan seluruh permainan yang telah dimainkan.

e. **RESET HISTORY**

RESET HISTORY merupakan command yang digunakan untuk menghapus semua history permainan yang dimainkan

Spesifikasi Game

1. RNG

Dijelaskan pada [Spesifikasi Tugas Besar 1 IF2111 2022/2023](#).

2. Diner Dash

Dijelaskan pada di [Spesifikasi Tugas Besar 1 IF2111 2022/2023](#).

3. Hangman

BNMO suka dengan *game* yang melibatkan tebak-tebakan kata sehingga ia membuat game yang terinspirasi dari game [Hangman](#). Berikut adalah spesifikasi game tersebut:

- Pada awal permainan program menentukan sebuah kata yang harus ditebak oleh pemain. **List kata dibebaskan kepada mahasiswa**. Boleh

ditentukan di *code*. Kemudian, pemain diberikan 10 kesempatan untuk melakukan penembakan huruf yang tidak terdapat dalam kata.

- Pada setiap giliran, pemain menebak satu huruf yang terdapat pada kata tersebut. Apabila huruf tebakan terdapat dalam kata, maka huruf yang sudah tertebak akan ditampilkan pada layar. Apabila salah, maka pemain kehilangan satu kesempatan. Pemain tidak boleh menebak huruf yang sudah ditebak sebelumnya pada kata yang sama.
- Apabila pemain berhasil menebak suatu kata, maka pemain tersebut diberikan **poin sesuai dengan panjang kata yang berhasil ditebak**, kemudian program memberikan kata baru yang harus ditebak oleh pemain dengan jumlah kesempatan yang tersisa.
- Permainan akan berlanjut hingga pemain kehabisan kesempatan untuk menebak huruf yang salah.

4. Tower of Hanoi

BNMO melihat Finn dan Jake sedang bermain Tower of Hanoi secara langsung, dengan adanya 3 tiang, dapat disebutkan sebagai tiang A, tiang B, dan tiang C dan posisinya terurut dari kiri ke kanan. Pada tiang A, terdapat 5 piringan, dengan piringan **paling bawah** merupakan piringan yang **paling besar** dan piringan **paling atas** merupakan piringan yang **paling kecil**. Cara bermainnya mudah, yaitu kelima piringan tersebut harus dipindahkan ke tiang C dengan posisi yang sama (piringan paling bawah merupakan piringan yang paling besar dan piringan paling atas merupakan piringan yang paling kecil), dengan peraturannya adalah piringan yang di bawah tidak boleh lebih ~~besar~~ **kecil** daripada piringan yang ada di atasnya. BNMO ingin membuat permainan ini dan agar lebih mengerti mengenai permainan ini, BNMO melihat panduannya di [The Enchiridion](#). Setelah itu, BNMO ingin melakukan modifikasi permainan ini:

- Jumlah piringan hanya 5 saja untuk permainan ini.
- Piringan direpresentasikan sebagai gambar piringan dengan *, dengan piringan paling besar adalah 9 dan balok silinder paling kecil adalah 1.
- Skor untuk permainan ini tergantung dengan seberapa optimal langkah dari pemain (dengan langkah paling optimalnya adalah 31) dan skor maksimalnya adalah 10 (Cara perhitungan skornya

dibebaskan, yang terpenting adalah jika langkahnya 31, skornya adalah 10).

5. Snake on Meteor

BNMO memiliki sebuah *game* andalannya yang menjadi daya tarik bagi para pemainnya, yaitu *snake on meteor*. Singkatnya, game ini mirip dengan *game snake* yang ada pada berbagai konsol lama, tetapi dipersulit dengan adanya kehadiran meteor yang dapat mengenai *snake* tersebut. Dampak yang didapat oleh pemain apabila *snake* terkena serangan meteor tersebut adalah panjang *snake* akan berkurang sebanyak 1 unit.

Untuk pemahaman spek tubes, kosakata ini akan digunakan:

1. **Kepala:** Bagian pertama dari *snake* (*hint: head pada linked list*)
2. **Badan:** Bagian yang bukan pertama dan terakhir dari *snake* (*hint: bagian yang ditunjuk oleh head dan terus berkait hingga menunjuk pada tail linked list*)
3. **Ekor:** Bagian terakhir dari *snake* (*hint: tail pada linked list*)

Berikut ini merupakan spesifikasi yang lebih *detail* terkait game *snake on meteor*:

- **Dimensi Peta:** Dimensi peta adalah 5x5 unit dengan <0,0> merupakan sisi kiri atas dan <4,4> sisi kanan bawah.
- **Panjang snake:** Panjang awal dari *snake* adalah 3 unit. Kepala dari *snake* di-*random* pada sebuah titik dan 2 anggota badan yang berurut menurun dengan prioritas horizontal yang sama. Apabila badan menabrak dinding, maka akan berurut menurun dengan prioritas vertical yang sama.
- **Makanan:** Makanan akan diberikan secara random pada sebuah titik <x,y> (ditandai dengan huruf **o**). Lokasi munculnya makanan tidak mungkin berada di titik yang sama dengan komponen tubuh *snake*. Jika berhasil dimakan oleh *snake*, maka *snake* akan langsung bertambah panjang ekornya sebanyak 1 unit dan makanan baru akan di-*random* lagi pada sebuah titik <x,y>

Proses pertambahan tail adalah sebagai berikut:

- a. **Secara umum**, ekor *snake* akan bertambah pada titik ordinat yang sama dengan tail, tetapi dengan titik koordinat satu sebelum tailnya (**Apabila tail berada pada titik $\langle x,y \rangle$, maka penambahan tail akan dilakukan pada titik $\langle x-1,y \rangle$**).
 - b. **Apabila kasus a tidak mungkin** (misalkan karena tail berada pada titik $\langle 0,1 \rangle$ dan tidak memungkinkan ada nilai titik $\langle -1,1 \rangle$), maka penambahan akan dilakukan pada titik ordinat satu sebelum ekor *snake* sekarang, tetapi pada titik koordinat yang sama (**Apabila tail berada pada titik $\langle x,y \rangle$, maka penambahan tail akan dilakukan pada titik $\langle x,y-1 \rangle$**).
 - c. **Apabila kasus b tidak mungkin** (misalkan karena ekor berada pada titik $\langle 0,0 \rangle$ dan tidak memungkinkan adanya nilai $\langle -1,0 \rangle$ ataupun $\langle 0,-1 \rangle$), maka penambahan akan dilakukan pada titik ordinat satu setelah ekor *snake* sekarang, tetapi pada titik koordinat yang sama (**Apabila tail berada pada titik $\langle x,y \rangle$, maka penambahan tail akan dilakukan pada titik $\langle x,y+1 \rangle$**).
 - d. **Apabila kasus a,b dan c tidak mungkin** (misalkan karena ekor berada pada titik $\langle 0,0 \rangle$ dan tidak memungkinkan adanya nilai $\langle -1,0 \rangle$ ataupun $\langle 0,-1 \rangle$ serta terdapat anggota tubuh pada titik $\langle 0,1 \rangle$), maka penambahan akan dilakukan pada titik ordinat yang sama dengan ekor *snake* sekarang, tetapi pada titik koordinat yang bertambah satu (**Apabila tail berada pada titik $\langle x,y \rangle$, maka penambahan tail akan dilakukan pada titik $\langle x+1,y \rangle$**).
 - e. **Apabila kasus a,b,c,d tidak mungkin** (misalkan ekor berada pada titik $\langle 2,2 \rangle$ dan terdapat anggota tubuh di titik $\langle 1,2 \rangle, \langle 2,1 \rangle, \langle 2,3 \rangle$ dan $\langle 3,2 \rangle$) maka game akan berakhir
- **Cara bergerak:** *Game* setiap putarannya akan meminta pemain untuk memasukkan huruf 'a', 'w', 's' atau 'd' untuk menggerakkan kepala *snake* (*game* akan meminta *re-input* apabila masukan selain huruf tersebut. Spesifikasi *lowercase* atau *uppercase* dibebaskan kepada kalian). Huruf 'a' untuk menggerakkan kepala ke kiri, 'w' untuk menggerakkan kepala ke atas, 's' untuk menggerakkan kepala ke bawah dan 'd' untuk menggerakkan kepala ke kanan. Setiap pergerakan akan menambahkan nilai koordinat/ordinat kepala *snake* (tergantung *input*

yang diberikan) sebanyak 1 unit. **Posisi pergerakan anggota tubuh akan mengikuti posisi anggota tubuh sebelumnya. Kepala *snake* tidak mungkin bergerak ke anggota tubuhnya sendiri(game akan meminta input ulang apabila hal tersebut terjadi)**

Contoh:

- a. Turn 0: Kepala *snake* berada pada titik $\langle 4,2 \rangle$ (titik H) dengan badannya berada pada $\langle 3,2 \rangle$ (titik 1) dan $\langle 2,2 \rangle$ (titik 2) secara berurutan (maka *snake* memiliki urutan $\langle 4,2 \rangle, \langle 3,2 \rangle, \langle 2,2 \rangle$)
 - b. Turn 1: Pemain memberikan masukan berupa 'w'. Posisi kepala *snake* akan berada pada titik $\langle 4,1 \rangle$. Sekarang bagian badan akan berpindah, maka anggota badan pada $\langle 3,2 \rangle$ akan berpindah ke $\langle 4,2 \rangle$ (mengikuti posisi head pada turn sebelumnya) dan anggota badan pada $\langle 2,2 \rangle$ akan berpindah ke $\langle 3,2 \rangle$ (mengikuti posisi titik 1 pada turn sebelumnya)
 - c. Turn 2: Pemain memberikan masukan berupa 'a'. Posisi kepala *snake* akan berada pada titik $\langle 3,1 \rangle$ dan anggota badan akan berada pada $\langle 4,1 \rangle$ dan $\langle 4,2 \rangle$
- **Meteor:** Setiap putaran setelah permainan berhasil digenerate (turn > 1), 1 meteor akan di-*random* pada titik tertentu (ditandai dengan huruf **m**). Apabila salah satu bagian dari *snake* terkena meteor, maka bagian tersebut akan dihapus dari *snake* dan panjang dari *snake* akan berkurang sebanyak 1. Apabila komponen dari *snake* (kepala/badan/ekor) terkena meteor (akan disebut *hit* untuk mempermudah pemahaman kalian), maka bagian badan sebelum *hit* akan tersambung dengan bagian badan setelah *hit*. Setelah terkena *hit*, ada kemungkinan badan *snake* berada di koordinat yang saling diagonal. Selanjutnya, **makanan tidak dapat muncul di titik ini dan kepala *snake* juga tidak bisa mengunjungi titik ini di turn selanjutnya.**
- **Kondisi Menang:** Tidak terdapat kondisi menang secara khusus. *Game* berakhir ketika terjadi kekalahan. Pada akhir game, satu unit pada komponen *snake* akan dikonversi menjadi 2 *point*.

Contoh: Pemain kalah dengan panjang akhir *snake* 10 unit, maka *score* yang didapat ialah 20 *point*

- **Kondisi Kalah:** Terdapat beberapa kondisi kekalahan dari *game* ini, yaitu
 1. Seluruh komponen *snake* (kepala, badan, ekor) terkena meteor → panjang *snake* adalah **0**
 2. Kepala dari *snake* terkena meteor → panjang *snake* adalah **panjang badan**
 3. Ekor baru tidak dapat di-*spawn* karena tidak dapat area di kiri, atas, bawah ataupun kanan ekor lama → panjang *snake* adalah **panjang badan sebelum ekor baru ditambahkan**
- **Contoh Permainan:Catatan:**
 1. Perhatikan dan validasi *input* gerak dari *snake*
Contoh: Kepala dari *snake* berada pada titik <4,3> dan badannya <3,3>,<2,3> (posisi *snake* secara berurutan: <4,3>,<3,3>,<2,3>). Maka pemain tidak dapat memberikan *input* 'a' pada gerak *snake*, karena akan mengakibatkan kepala *snake* bergerak menuju badan-nya sendiri
 2. Penggambaran peta dibebaskan, boleh menggunakan tabel dengan sekat-sekat atau tabel tanpa sekat-sekat
 3. Gunakan prinsip “apabila ekor tidak dapat *spawn* di-kiri, maka akan dilakukan *spawn* di atas. Apabila di kiri dan di atas tidak mungkin, *spawn* ekor di bawah. Apabila tidak mungkin *spawn* di kiri, atas dan bawah, maka lakukan *spawn* di kanan. Apabila tidak memungkinkan untuk *spawn*, maka akan *game over*”
 4. Peta **tidak harus** ‘tembus’ atau muncul pada sisi sebaliknya apabila dilewati oleh *snake*.
Contoh: Dengan dimensi 5x5, maka titik minimal dari peta adalah <0,0> dan titik maksimal dari peta adalah <4,4>. Apabila kepala berada pada titik <0,0> dan dilakukan *input* ‘a’, maka pada putaran berikutnya kepala tidak harus berada pada titik <4,0>

Kalian dibebaskan untuk menangani kasus tersebut, apakah kalian ingin melakukan validasi *input* kembali atau *snake* akan digerakan secara *random* oleh sistem/lain-lainnya.

6. Game tambahan/ Buatan pemain

Game buatan pemain yang dibuat menggunakan command CREATE GAME akan langsung selesai dan masuk ke tahap game over dengan skor akhir berupa integer random.

- Daftar ADT yang Digunakan

Anda diwajibkan menggunakan ADT di bawah ini. Selain itu, Anda dapat pula menggunakan ADT lain, namun cantumkan analisis alasan kenapa menggunakan ADT tersebut pada laporan.

- ADT Stack

ADT ini digunakan untuk merepresentasikan urutan dari permainan yang telah dimainkan, dengan TOP adalah permainan yang baru saja dimainkan dan digunakan untuk permainan Tower of Hanoi.

- ADT Set & Map

ADT Set digunakan untuk menyimpan nama di setiap *game* sehingga memastikan tidak ada nama yang duplikat. Gunakan set yang berbeda di setiap *game* untuk memudahkan pencatatan.

ADT Map digunakan untuk menyimpan skor untuk setiap nama. Gunakan Map yang berbeda di setiap *game* untuk memudahkan pencatatan. Implementasi map menggunakan list dengan elemennya berupa struct yang berisi key dan value.

- ADT Linked List

ADT ini digunakan untuk mengimplementasikan game *snake on meteor*. Tipe data yang ditampung dalam *linked list* berupa *point* yang menyimpan nilai $\langle x,y \rangle$. *Linked list* minimal digunakan pada representasi lokasi badan *snake*.

- Bonus

Pada tugas besar ini, terdapat beberapa fitur yang berupa bonus. Fitur-fitur ini tidak wajib untuk dikerjakan dan bobotnya lebih kecil dibanding fitur utama.

1. Game custom menggunakan ADT Tree

Mahasiswa dibebaskan menambahkan jenis permainan dengan mengimplementasikan ADT Tree. Permainan harus memiliki output berupa score yang bertipe integer. Permainan tidak boleh menggunakan library selain dari library yang telah ditentukan.

2. Penambahan fitur pada game Hangman

- **Penambahan in-game dictionary.** Terdapat 2 menu ketika memainkan permainan Hangman, yaitu bermain langsung atau menambah kata-kata ke dalam kamus/ list kata yang berada dalam daftar kata.
- **List kata dibaca dari file.** Di awal permainan, program membaca list kata dari file. Nama file dibebaskan oleh program (tidak di-*input* oleh pengguna). Format *file* juga dibebaskan asalkan menunjukkan daftar kata yang akan dibaca. Setelah permainan selesai/*game over*, *list* kata disimpan kembali ke *file* karena *list* kata mungkin saja bertambah.

3. Penambahan fitur pada game Tower of Hanoi

- **Opsi jumlah piringan.** Sebelum permainan dimulai, akan diminta opsi jumlah piringan yang digunakan. Kemudian, skor yang diperoleh akan disesuaikan dengan jumlah piringan pada game (Pemain dengan jumlah piringan yang lebih sedikit akan memperoleh nilai maksimal lebih rendah daripada pemain dengan jumlah piringan yang lebih banyak).

4. Penambahan fitur pada game Snake on Meteor

- **Penambahan obstacle.** Ketika Kepala dari *snake* mengenai obstacle, maka permainan berakhir. *Obstacle* muncul di awal permainan dan tidak dapat ditembus oleh *snake*. Selain itu, makanan juga tidak dapat muncul pada titik yang memiliki *obstacle*.
- **Menghubungkan sisi peta yang berseberangan.** Ketika kepala *snake* melewati sisi atas peta, maka kepala *snake* akan muncul dari sisi

bawah peta. Hal yang sama berlaku ketika kepala *snake* melewati sisi kiri/ kanan peta, maka kepala snake akan muncul dari sisi yang berlawanan. GIF dibawah merupakan visualisasi poin spek ini. Anggap saja ukuran map 5 x 5.




9.2 Notulen Rapat




**Form Asistensi Tugas Besar
IF2110/Algoritma dan Struktur Data
Sem. 1 2022/2023**

No. Kelompok/Kelas : 1 / K03
Nama Kelompok : Kelompok Satu
Anggota Kelompok (Nama/NIM) :
1. Wan Aufa Aziz / 18221001
2. Athira Dhyannis / 18221022
3. Dhafin Ghalib Lukman Hakim / 18221023
4. Syasya Umaira / 18221026
5. Muhammad Mumtaz / 18221029






Asisten Pembimbing : Graciella Valeska Liander


Asistensi I

Tanggal : 25 November 2022	Catatan Asistensi: Update progress dulu Udah nambahin ADT-ADT yang diperluin lagi, set belum dipake, untuk game baru nyoba di tower of hanoi, hangman sama snake msh on progress, kalo di consol udh nambah buat reset sama history penamaan functionnya usahain konsisten, di emptyset coba diseragamin kayak tulisan kapital dan sebagainya, tapi itu minor banget kok, sisanya oke, untuk yang set boleh gak pake map aja masih aku diskusiin sama asisten asisten lain kendala lain? aku mau nanya ttg snake, konsultasiin ide, di speknya make list, buat ngeprint ui nya itu aku bikin matriks of integer gitu aman ga ya kak? representasi obstacles, badan ular gtgt aku taronya di matriks gt? boleh, yang dipilih sekarang udah oke kok dia kurang konsisten di badannya gt di panjangnya kayak ngurang gt trs dibawah kiri kanan atas ternyata 0,0 disebelah kiri atas bukan dari bawah iya 0,0 dari kiri atas untuk visualisasinya itu kondisi matinya itu cuman 3 kan kalo kepala kena, badan udah abis, atau ekornya
Tempat : Zoom	
Kehadiran Anggota Kelompok: No NIM Tanda tangan 1 18221001  2 18221022  3 18221023 	

<p>4 18221026</p>  <p>5 18221029</p> 	<p>gabisa kemana mana kalo ularnya kena ujung ujungnya diitung mati gak? iya kalo ga bikin bonus</p> <p>Ngebedain naikinnya gimana itu bingung, pas aku buat, ada yang nanitnya jadi negatif itu ya mungkin instead ngurang bs di kurangin aja</p> <p>isi branchnya ga update walaupun udah di gitpull jadi waktu nanti udh di push nanti isinya masih harus dibenerin satu satu gitu sesuai yang udh di update</p> <p>pertama, selalu pull dulu dr main, kerjain pas mau ngepush pull lagi, selama ngerjain kode ada yg nge push</p> <p>untuk semua game harus ada nilainya boleh bikin cheat command, jd isinya kayak mungkin bisa langsung selesai, jangan lupa cicil laporan sama form asistensi!!</p> <p>algoritma menarik, kl mau load atau save kita pindah mode satu atau dua dulu itu bisa di masukan ke algoritma menarik</p> <p>data test bisa liat ke spek, commandnya apa ekspetasi dari program kita apa (ga harus ikut spek) test script commandnya apa, realitanya yang keluar apa dari program kita</p>
	<p>Tanda Tangan Asisten:</p> 

Asistensi II

Tanggal : 2 Desember 2022	Catatan Asistensi: Progres : Untuk progress krn udah h-1 jam juga, udah cukup aman semua, ADT sudah lengkap dan nambahin yang baru kayak list matriks map buat game juga sudah aman hangman snake on meteor sama tower of hanoi di console jg udh aman buat fungsi mainnya juga masih sama di read me jg sudah buat 2 sistem windows sm linux sebenarnya linux sm windows sama aja sama sama pake gcc, command yg dipake sama persis cmn kl linux kl gcc di jalanin di sistemnya Buat di snake on meteor ini aku gatau kenapa ada bug pas nambah ekor kadang diatas kadang engga itu gimana ya kak? setiap jalan makannya pindah ga? enggak kalo prosedur pas makan yg mana? one turn dia ngecek kayak badan belakangnya ngikutin badan depannya, ekor ngikutin sampe p nya di kepala dia bakal muncul kl inputan sudah pasti valid logikanya udah bener, tapi kalian sudah pernah ngecek satu satu gak dia waktu jalan makan dimana gt? sebenarnya bisa langsung diliat disini sih kak coba di bug dari codenya coba ke codenya lalu print dulu info p, get elemen matriks, absis elemen, dan ordinat elemen. buat liat jalannya gimana dulu. mungkin pas dia makan, tapi matriksnya itu belum berubah apakah aku harus selalu insert biar matriksnya selalu update jadi coba tambahin update lagi matriksnya
Tempat : Zoom	
Kehadiran Anggota Kelompok:	
No NIM Tanda tangan 1 18221001 	
2 18221022 	
3 18221023 	
4 18221026  5 18221029 	

	Tanda Tangan Asisten: 

9.3 Log Activity Anggota Kelompok

Timeline Kelompok

No	Waktu	Keterangan
1	Minggu, 20 November 2022	<ul style="list-style-type: none"> Meet kelompok untuk pembagian tugas
2	Rabu, 23 November 2022	<ul style="list-style-type: none"> Meet membahas perkembangan dari masing masing tugas
3	Jumat, 25 November 2022	<ul style="list-style-type: none"> Asistensi 1 Membahas hasil asistensi 1
5	Selasa, 29 November 2022	<ul style="list-style-type: none"> Memulai pengerjaan laporan
6	Jumat, 2 Desember 2022	<ul style="list-style-type: none"> Asistensi 2 Penyelesaian laporan Melakukan <i>finishing</i> pada code program