

LAPORAN TUGAS BESAR

IF2111 Algoritma dan Struktur Data


Bimono

Dipersiapkan oleh:

Kelompok 4

I Dewa Made Manu Pradnyana	18221047
Laurentia Kayleen Christopher	18221053
Felisa Aidadora Darmawan	18221137
Ananda Abdul Hafizh	18221167
Hans Stephano Edbert N	18221171

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung
Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		<i>IF2111-TB2-04</i>		39
		<i>Revisi</i>	0	02 Desember 2022

Daftar Isi

1 Ringkasan	3
2 Penjelasan Tambahan Spesifikasi Tugas	
2.1 Penambahan Fitur pada Game Hangman	3
2.2 Penambahan Fitur pada Game Tower of Hanoi	4
2.3 Penambahan Fitur pada Game Snake on Meteor	4
3 Struktur Data	
3.1 ADT Stack	4
3.2 ADT Set	5
3.3 ADT Map	5
3.4 ADT Linked List	6
3.5 ADT Array	6
3.6 ADT Mesin Kata	7
4 Program Utama	
4.1 Inisialisasi dan Konfigurasi	7
4.2 Command LOAD <filename>	8
4.3 Command SAVE <filename>	8
4.4 Command CREATEGAME	8
4.5 Command DELETGAME	8
4.6 Command PLAYGAME	9
4.7 Command HELP	9
4.8 Command SCOREBOARD	9
4.9 Command RESET SCOREBOARD	10
4.10 Command HISTORY <n>	10
4.11 Command RESET HISTORY	10
4.12 Game Hangman	11
4.13 Game Tower Of Hanoi	11
4.14 Game Snake On Meteor	12
5 Data Test	
5.1 LOAD <filename>	12
5.2 CREATEGAME	13
5.3 DELETGAME	14
5.4 PLAYGAME	14
5.5 HELP	15
5.6 SAVE <namafile>	16
5.7 Command SCOREBOARD	16
5.8 Command RESET SCOREBOARD	17
5.9 Command HISTORY <n>	18
5.10 Command RESET HISTORY	20
5.11 Game Hangman	20
5.12 Game Tower Of Hanoi	24
5.13 Game Snake On Meteor	26
6 Test Script	30
7 Pembagian Kerja dalam Kelompok	33
8 Lampiran	
8.1 Deskripsi Tugas Besar 2	35
8.2 Notulen Rapat	35
8.3 Log Activity Anggota Kelompok	40

1 Ringkasan

Tugas ini melibatkan sebuah robot video game console bernama BNMO. Karena beberapa hal, BNMO sempat rusak, namun sudah diperbaiki—tentu saja, ada beberapa hal yang kurang sejak perbaikan tersebut. Fitur-fitur yang ada pada BNMO harus dikembalikan lagi. Oleh karena itu, kelompok kami berusaha untuk melengkapi kembali fitur-fitur BNMO agar bisa dimainkan lagi. Namun, hal yang berbeda adalah kita harus menambahkan fitur-fitur baru yang akan meng-*improve* BNMO menjadi lebih keren lagi~

Laporan ini seperti biasa terdiri atas beberapa bagian utama. Setelah ringkasan ini, akan ada penjelasan tambahan mengenai spesifikasi tugas yang perlu diketahui oleh pembaca, penjelasan struktur data atau ADT yang digunakan, penjelasan program utama yang kami buat, hasil dari *data test* yang dilakukan pada main program kami, skenario *test* yang kami gunakan seperti ekspektasi output dari setiap command di program, pembagian kerja kelompok kami untuk melihat kontribusi dan siapa saja yang melakukannya, serta lampiran penting yang dibutuhkan untuk pengertian lebih lanjut mengenai laporan ini.

Sebagai kesimpulan dan disertai informasi lebih detail, BNMO dibuat dengan menggunakan bahasa C dengan menggunakan beberapa ADT yang sudah dipelajari di mata kuliah IF2111 Algoritma dan Struktur Data. ADT yang digunakan antara lain adalah ADT array, ADT mesin karakter, ADT mesin kata, ADT queue, ADT stack, ADT linked list, dan ADT tree.

2 Penjelasan Tambahan Spesifikasi Tugas

Pada Tugas Besar 2, terdapat fitur tambahan sebagai bonus pada tugas ini, yaitu menambahkan fitur Penambahan in-game dictionary dan List kata dibaca dari file pada game Hangman, menambahkan opsi jumlah piringan pada game Tower of Hanoi, dan Penambahan obstacle dan menghubungkan sisi peta yang berseberangan pada Snake on Meteor.

2.1 Penambahan Fitur pada Game HANGMAN

Penambahan in-game dictionary : Terdapat 3 menu ketika memainkan permainan Hangman, yaitu bermain langsung (PLAYGAME), menambah kata-kata ke dalam kamus dan menambah soal ke dalam listSoal (ADDWORD), dan keluar dari permainan (QUIT). Bonus tersebut dikerjakan dengan membuat prosedur *menuawal*.

List kata dibaca dari file : Di awal permainan, program membaca list kata dari file. Nama file untuk load adalah hangman.txt. Untuk LOAD dan SAVE list kata dari file dikerjakan dengan membuat prosedur *ListOfWord* (untuk load list) dan *savefile*.

2.2 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). Bonus tersebut dikerjakan dengan meminta input jumlah piringan pada main serta disimpan ke variabel tumpukan yang nanti akan menentukan banyaknya piringan pada tower.

2.3 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. Bonus tersebut dikerjakan dengan menggunakan prosedur `GameSetSnake` yang meminta inputan luas arena dan banyak obstacle yang digunakan dan setelah itu menambahkan obstacle pada matriks arena

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. Bonus tersebut dikerjakan dengan memanfaatkan `next_row`, `next_cols`, `prev_cols`, dan `prev_row` pada struktur data `MatriksNode` sehingga dapat menghubungkan sisi peta yang berseberangan.

3 Struktur Data (ADT)

3.1 ADT Stack

ADT *stack* pada file `stack.h` digunakan untuk menyelesaikan persoalan terkait dengan sejarah atau *history* dari games yang telah dimainkan oleh *user*. Sistem ADT *stack* adalah yang teratas merupakan game terakhir yang dimainkan, maka sangat tepat menggunakan ADT *stack* ini. ADT *stack* akan digunakan untuk menyimpan nama-nama game yang telah dimainkan oleh *user*. Game yang terakhir dimainkan adalah game yang namanya akan ditempatkan di urutan paling atas dari *stack*. Terdapat beberapa fungsi primitif yang didefinisikan dalam ADT ini. Pertama, ada ***CreateEmptyStack***, sebuah prosedur yang digunakan untuk membuat *stack* kosong. Lalu, ada ***isStackEmpty***, fungsi yang akan menampilkan boolean berupa “true” apabila *stack* kosong. Selanjutnya, ada ***isStackFull***, fungsi yang sebaliknya akan menampilkan boolean berupa “true” apabila *stack* penuh. Prosedur ***Push*** adalah prosedur yang akan menambahkan elemen

pada *stack* dengan nilai yang kita inginkan. Prosedur **Pop** adalah prosedur yang akan menghapus/mengeluarkan elemen teratas pada *stack*.

3.2 ADT Set

ADT *Set* merupakan implementasi model himpunan matematika sehingga keseluruhan elemen dalam *Set* merupakan elemen yang unik. ADT *Set* dalam sistem program digunakan untuk menyimpan *key value* dari *Key* dalam ADT *Map*, yakni nama *player*, yang digunakan dalam fungsi *SCOREBOARD*. ADT ini didefinisikan dan diimplementasikan dalam file **set.h** dan **set.c**. Elemen yang disimpan bertipe *Word* yang diimplementasikan dalam file *mesinkata.h*. Tipe *Set* dalam ADT ini memiliki stuktur *array of Word: PlayerName* berukuran 50 yang digunakan untuk menyimpan nama dari para *player*, serta *address: Count* yang menunjukkan banyak elemen efektif dalam array. Selain tipe *Set*, terdapat juga tipe bentukan *ListOfSet* yang tersusun atas *array of Set: GameSet* berukuran 20 yang digunakan untuk menyimpan himpunan dari nama-nama *player* untuk setiap *game* sehingga indeks *GameSet* yang berbeda menunjukkan himpunan nama *player* dari *game* yang berbeda, serta *address: Num* yang menunjukkan banyak elemen aktif dalam *ListOfSet*.

Terdapat beberapa fungsi primitif yang didefinisikan dalam ADT ini. **CreateEmptySet** merupakan prosedur yang digunakan untuk membuat suatu *Set* menjadi kosong atau *Count* dalam *Set* tersebut bernilai 0. **IsSetEmpty** merupakan fungsi yang mengembalikan tipe *boolean true* jika *Set* kosong dan *false* jika tidak. **IsSetFull** merupakan fungsi yang mengembalikan tipe *boolean true* jika *Set* penuh atau bernilai *MaxSetEl* (20). **InsertSetEl** merupakan prosedur yang digunakan untuk memasukkan suatu elemen baru ke dalam *Set*. Prosedur ini menjamin keunikan tiap elemen dalam *Set*. **DeleteSetEl** merupakan suatu prosedur yang digunakan untuk menghapus suatu elemen tertentu dalam *Set*. **IsSetMember** merupakan fungsi yang mengembalikan tipe *boolean true* jika suatu elemen masukan terdapat di dalam suatu *Set* dan *false* jika tidak. **CreateEmptySetList** merupakan suatu prosedur yang membuat suatu *ListOfSet* kosong, yakni dengan *Num* bernilai 6 yang mengindikasikan *ListOfPlayer* tidak berisikan *Set of Players* dari *game* yang dibuat oleh *user*. **SortSetByMap** merupakan prosedur *sorting* dari suatu *Set* yang melihat keterurutan dari *Map* masukan sehingga dibutuhkan *import map.h*.

3.3 ADT Map

ADT *Map* merupakan himpunan pasangan / *binding* $\langle \text{key}, \text{value} \rangle$, dengan nilai *key* bersifat unik di dalam himpunan tersebut. ADT ini didefinisikan dan diimplementasikan dalam file *map.h* dan *map.c* ADT *Map* digunakan dalam fungsi *SCOREBOARD* dengan *key* berisikan nama *player* dan *value* yang berisikan skor dari *game* yang baru saja dimainkan. Di dalam *Map Scoreboard* sistem, tiap *binding* terurut mengecil berdasarkan *value* atau skor pemain. Namun,

terkhusus dalam game *custom* MARVEL SNAP, tidak dilakukan *sorting* berdasarkan *value* (penjelasan lebih lanjut dielaborasikan dalam bagian [4.1 Command SCOREBOARD](#)).

Tipe *Map* dalam ADT ini tersusun atas *array of mapinfo: Elements* berukuran 100, serta *address: Count* yang menunjukkan banyak elemen efektif dalam *Map*. Tipe *mapinfo* tersusun atas *keytype: Key* dengan *keytype* bertipe *Word* dan *valuetype: Value* dengan *valuetype* bertipe *integer*. Selain itu, dibentuk juga tipe *ListOfMap* yang tersusun atas *array of Map: board* berukuran 20 yang tiap indeksinya menyimpan *Scoreboard (Map)* dari game yang berbeda, serta *address: Num* yang menunjukkan banyak elemen efektif dalam *ListOfMap*.

Terdapat beberapa fungsi primitif yang didefinisikan dalam ADT ini. ***CreateEmptyMap*** merupakan prosedur yang digunakan untuk membuat suatu *Map* menjadi kosong, yakni dengan *Count* bernilai 0. ***IsMapEmpty*** merupakan fungsi yang mengembalikan nilai *boolean true* apabila *Map* kosong dan *false* apabila tidak. ***IsMapFull*** merupakan fungsi yang mengembalikan nilai *boolean true* apabila *Map* penuh, yakni dengan *Count* bernilai *MaxMapEl* (100), dan *false* apabila tidak. ***Value*** merupakan fungsi yang mengembalikan nilai *value* dari suatu *Map* dengan masukan sebuah *Key*. ***MapValIns*** merupakan prosedur yang digunakan untuk memasukkan *binding* baru. ***MapValDel*** merupakan prosedur yang digunakan untuk menghapus *binding* tertentu berdasarkan masukan *Key*. ***IsMapMember*** merupakan fungsi yang mengembalikan nilai *boolean true* apabila masukan *Key* terdapat dalam *Map* dan *false* apabila tidak. ***copyMap*** merupakan prosedur yang digunakan untuk menyalin suatu *Map* ke *Map* lainnya. ***CreateEmptyMapList*** merupakan prosedur yang digunakan untuk mengosongkan suatu *ListOfMap*, yakni dengan *Num* bernilai *default* 6 karena pada konfigurasi awal sistem sudah pasti terdapat 6 game sehingga dibutuhkan 6 *Scoreboard*. ***SortByVal*** merupakan prosedur yang digunakan untuk melakukan *sorting* terhadap *Map* berdasarkan *Value* terurut mengecil.

3.4 ADT Linked List

ADT linked list pada file *linkedlist.h* digunakan secara khusus untuk game snack on meteor saja sehingga implementasi ADT yang kami gunakan pun dibuat khusus untuk game ini saja. Struktur linked list yang kami gunakan merupakan matrix linked list dengan masing - masing nodenya (*typedef struct MatriksNode*) terdiri dari info bertipe *char**, lalu *next_row*, *prev_row*, *next_col*, dan *prev row* bertipe *addressNode* untuk mengakses baris dan kolom sesudah dan sebelum lokasi saat itu pada matriks linked list, lalu *next* dan *prev* bertipe *addressNode* untuk mengakses elemen sesudah dan sebelum dari matriks linked list (digunakan untuk mengakses badan sebelum dan sesudah pada snake-nya). *MatriksNode* dapat dibuat dengan menggunakan prosedur ***create_node***. Kemudian terdapat struktur *List* yang merupakan kumpulan dari *MatriksNode*. Pada struktur *List* terdiri dari Zero (untuk mengakses matriks pada baris dan kolom 1) , First (Untuk mengakses kepala dari snake) ,dan Last (Untuk mengakses ekor dari snake) bertipe *addressNode*. *List* dapat dibentuk menggunakan prosedur ***create_linked_list_matrix*** untuk membuat matriks linked list dan ***create_snake*** untuk membuat

snake pada matriks. Kemudian terdapat beberapa fungsi dan prosedur yang digunakan untuk pembuatan game snake on meteor, yaitu prosedur *movesnake* untuk membuat ular bergerak pada matriks linked list, kemudian untuk makanan dibuat prosedur *MakeFood* untuk menentukan lokasi makanan pada matriks secara random, prosedur *SummonFood* untuk memasukan makanan ke matriks, dan fungsi *isFood* untuk memeriksa apakah ada makanan pada matriks. Kemudian untuk meteor dibuat prosedur *MakeMeteor* untuk menentukan lokasi meteor pada matriks secara random, prosedur *SummonMeteor* untuk memasukan meteor ke matriks, fungsi *isNabrakMeteor* untuk memeriksa apakah ular menabrak meteor yang masih panas, dan prosedur *MeteorDisappear* untuk menghilangkan meteor yang sudah tidak panas pada matriks. Kemudian untuk obstacle dibuat prosedur *makeobstacle* untuk menentukan lokasi obstacle pada matriks secara random dan prosedur *summonobstacle* untuk memasukan meteor ke matriks. Kemudian untuk membentuk badan pada ular dibuat prosedur *makebody* dan untuk memeriksa apakah snake menabrak badan digunakan fungsi *isNabrakBody*. Kemudian untuk kondisi kalah dari game ini, yaitu menabrak obstacle diperiksa menggunakan fungsi *isNabrak* dan untuk kondisi snake tidak bisa bergerak kenapa pun karena kanan kiri depan dan belakangnya adalah badan snake, diperiksa menggunakan fungsi *isNotMoveable*.

3.5 ADT Array

ADT *Mesin Kata* yang didefinisikan dan direalisasikan dalam file *array.h* dan *array.c* merupakan ADT yang berperan dalam menyimpan sejumlah tipe *Word*. ADT ini sudah dipergunakan sejak Tugas Besar 1, tetapi terdapat sebuah penyesuaian yang harus dibuat pada Tugas Besar 2, yakni penambahan fungsi *isMemberArray* yang mengembalikan nilai *boolean true* ketika masukan tipe *Word* terdapat dalam suatu array *TabWord*.

3.6 ADT Mesin Kata

ADT *Mesin Kata* yang didefinisikan dan direalisasikan dalam file *mesinkata.h* dan *mesinkata.c* merupakan ADT yang wajib dimiliki dalam program ini karena ADT ini berperan dalam menerima dan membaca masukan, baik dari *keyboard* maupun dari *textfiles*. ADT ini sudah dipergunakan sejak Tugas Besar 1, tetapi terdapat beberapa penyesuaian yang harus dibuat serta penjelasan lebih lanjut guna memenuhi fitur-fitur baru yang terdapat dalam Tugas Besar 2.

Pada prosedur *COPYWORD*, dilakukan penyesuaian terhadap pembacaan *EOL (End Of Line)* untuk *OS Windows* (*\r\n*) dan *Linux* (*\n*) sehingga *textfile* dapat dibaca di kedua OS. Pada prosedur *COPYGAME*, dilakukan pengakuisisian isi dari *textfile* tanpa memerhatikan BLANK. Terdapat juga fungsi *isNumber* yang mengembalikan nilai *boolean true* apabila tipe *Word* yang dibaca merupakan suatu angka dan *false* jika tidak.

Beberapa penambahan yang dilakukan dalam ADT ini: ¹⁾prosedur *binSep* yang digunakan untuk melakukan pemisahan tipe *Word* secara *binary* pada suatu tipe *Word* berdasarkan masukan *separator* yang diberikan sehingga dihasilkan 2 tipe *Word* yang berbeda; ²⁾fungsi *strcompare*

yang digunakan untuk membandingkan dua tipe *string* dan mengembalikan *true* jika kedua *string* identik; ³⁾prosedur ***clear*** yang digunakan untuk membersihkan layar yang dapat berlaku untuk beberapa *OS*; dan ⁴⁾prosedur ***wordCopy*** yang digunakan untuk menyalin suatu tipe *Word* pada tipe *Word* lainnya.

4 Program Utama

4.1 Inisialisasi dan Konfigurasi

Pada Tugas Besar 2, *file config* memiliki isi yang sama, tetapi terdapat beberapa penambahan inisialisasi, yakni *History* bertipe *Stack*, *listPlayer* bertipe *ListOfSet*, dan *scoreBoard* yang bertipe *ListOfMap*. Sesuai penamaan, *History* digunakan untuk menyimpan riwayat permainan yang dimainkan oleh *user*. Sementara itu, *listPlayer* digunakan untuk menyimpan nama-nama *player* dari setiap game. Tiap indeks *Set* dalam *ListOfSet* menyimpan nama-nama *player* dalam papan skor sesuai indeks pada *TabWord listGame*. *ListOfMap scoreBoard* digunakan untuk menyimpan nama dan skor dari tiap *player* yang sudah memainkan game. *ListOfMap* tersebut terurut mengecil berdasarkan skor *player*, terkecuali untuk game MARVEL SNAP.

```
/* *** INISIALISASI *** */
Queue queueGame; CreateQueue(&queueGame);
TabWord listGame; MakeTabWord(&listGame);
TabWord listCommand; MakeTabWord(&listCommand);
Stack History; CreateEmptyStack(&History);
ListOfSet listPlayer; CreateEmptySetList(&listPlayer);
ListOfMap scoreBoard; CreateEmptyMapList(&scoreBoard);

boolean active = true;
boolean loaded = false;
```

Gambar 4.1.1: Penambahan Inisialisasi

4.2 Command LOAD <filename>

Pada Tugas Besar 2, terdapat perubahan terkait prosedur *LOAD* karena terdapat perubahan terhadap isi *save file* yang harus menyimpan *data* riwayat permainan dan scoreboard dari setiap game yang terdaftar dalam sistem. Prosedur *LOAD* kini harus meminta masukan *Stack History* untuk menyimpan riwayat permainan, *ListOfSet listPlayer* untuk menyimpan nama dari setiap pemain, dan *ListOfMap scoreBoard* untuk menyimpan skor dari setiap pemain sesuai nama mereka.

4.3 Command **SAVE** <filename>

Pada Tugas Besar 2, terdapat perubahan terkait prosedur *SAVE* karena terdapat perubahan terhadap isi *save file* yang harus menyimpan *data* riwayat permainan dan scoreboard dari setiap game yang terdaftar dalam sistem. Prosedur *SAVE* kini harus meminta masukan *Stack History* sebagai sumber riwayat permainan dan *ListOfMap scoreboard* sebagai sumber nama dari setiap pemain serta skor yang mereka dapatkan dalam game-game tertentu.

4.4 Command **CREATE GAME**

Pada Tugas Besar 2, terdapat perubahan terkait prosedur *CREATE GAME* karena terdapat fitur baru, yakni *scoreboard* (papan skor) yang menampilkan skor dari para pemain dari setiap gamenya. Prosedur *CREATE GAME* kini harus meminta argumen *ListOfSet* dan *ListOfMap* dan memodifikasi indeks efektif *ListOfSet* dan *ListOfMap* sehingga dihasilkan *Set* dan *Map* yang berlaku untuk game yang baru saja dibuat oleh *user*.

4.5 Command **DELETE GAME**

Pada Tugas Besar 2, terdapat perubahan terkait prosedur *DELETE GAME* karena terdapat fitur baru, yakni *scoreboard* (papan skor) yang menampilkan skor dari para pemain dari setiap gamenya. Prosedur *DELETE GAME* kini harus meminta argumen *ListOfSet* dan *ListOfMap* dan memodifikasi indeks efektif *ListOfSet* dan *ListOfMap* sehingga *Set* dan *Map* yang berlaku untuk game hasil buatan oleh *user* tidak ditampilkan dan digunakan.

4.6 Command **PLAY GAME**

Pada Tugas Besar 2, terdapat perubahan terkait prosedur *PLAY GAME* karena terdapat penambahan beberapa game baru, yakni HANGMAN, TOWER OF HANOI, dan SNAKE ON METEOR. Ketiga game baru tersebut berkedudukan di urutan ketiga hingga kelima di dalam *listGame*. Penambahan game ini mengharuskan prosedur *PLAY GAME* untuk memanggil fungsi utama game agar dapat dimainkan. Selain itu, prosedur ini meminta dan menerima skor dari game yang baru dimainkan serta nama dari *player* dengan memanfaatkan prosedur *UPDATESB* untuk memperbarui *scoreboard*.

```

void UPDATESB(int score, Set *gamePlayers, Map *playerScores, int whatGame)
{
    // Check apakah sudah berada dalam set
    printf("Masukkan Nama: ");
    STARTCMD(false);
    UPPER(&currentCommand);
    while (IsSetMember((*gamePlayers), currentCommand))
    {
        printf("Nama Sudah Terdaftar dalam SCOREBOARD!\n");
        printf("Harap masukkan Nama lainnya!\n");
        printf("Masukkan Nama: ");
        STARTCMD(false);
        UPPER(&currentCommand);
    }

    printf("Berhasil Menambahkan Data dalam SCOREBOARD!\n");
    InsertSetEl(gamePlayers, currentCommand);
    MapValIns(playerScores, currentCommand, score);
    if (whatGame != 6) // Kalo MARVELSNAP, jangan di Sort
    {
        SortByVal(playerScores);
        SortSetByMap(gamePlayers, (*playerScores));
    }
}

```

Gambar 4.6.1: Prosedur *UPDATES*

4.7 Command *HELP*

Pada Tugas Besar 2, terdapat perubahan terkait prosedur *HELP* karena terdapat penambahan beberapa fungsi baru. Beberapa command yang ditambahkan dalam prosedur *HELP* adalah *SCOREBOARD*, *RESET SCOREBOARD*, *HISTORY <n>*, dan *RESET HISTORY*.

4.8 Command *SCOREBOARD*

Ketika command *SCOREBOARD* dipanggil, program akan langsung memanggil fungsi *SCOREBOARD()* dengan argumen *ListOfMap Scoreboard*, *ListOfSet Players*, dan *TabWord ListGame*. Fungsi *SCOREBOARD()* akan melakukan *looping* sebanyak jumlah game yang terdaftar kemudian menampilkan nama *scoreboard* beserta judul game, tabel 2 kolom: NAMA dan SKOR, serta isi dari tabel yakni nama *player* dan skor yang ia dapatkan. Jika di dalam game tertentu, *Set* nama *player* kosong, maka ditampilkan “SCOREBOARD KOSONG” sebagai isi dari tabel. Jika *Set* yang sedang diakses merupakan *Set* untuk game MARVEL SNAP, maka ditampilkan urutan *Match* dan dua *player* yang bermain pada *match* tersebut beserta skor mereka, seperti pada *Scoreboard* game lainnya.

4.9 Command *RESET SCOREBOARD*

Ketika command *RESET SCOREBOARD* dipanggil, program akan langsung memanggil fungsi *RESETSB()* dengan argumen *ListOfMap scoreBoard*, *ListOfSet listPlayers*, dan *TabWord listGame*. Prosedur *RESETSB* akan menampilkan keseluruhan nama game secara terurut berdasarkan *listGame* serta pilihan “ALL” sebagai pilihan nomor 0. Kemudian, prosedur meminta masukan angka dari *user* dan memvalidasi masukan. Jika masukan tidak *valid*, maka ditampilkan *Error Message* terkait *Invalid Input* dan membatalkan *reset*. Jika masukan

valid, maka fungsi akan meminta konfirmasi penghapusan dari *user* berupa jawaban “YA” atau “TIDAK”. Jika masukan tidak *valid*, maka ditampilkan *Error Message* terkait *Invalid Input* dan membatalkan *reset*. Jika masukan *valid*, maka prosedur akan mengosongkan sebuah, keseluruhan, atau membatalkan *reset SCOREBOARD*.

4.10 Command HISTORY <n>

Ketika command HISTORY <n> (dengan n adalah jumlah nama *game* yang ingin ditampilkan sebagai urutan dari *game* yang telah dimainkan) dipanggil, program akan mengecek terlebih dahulu apakah angka yang dimasukkan angka yang positif dan *valid*. Jika tidak, maka program akan menampilkan *error message* “Masukan Angka Tidak Valid.” dan “Mohon masukkan angka yang bernilai positif!”. Apabila *valid*, maka akan ditampilkan “Berikut adalah daftar *game* yang telah dimainkan” dan akan ditunjukkan urutan *game* yang telah dimainkan sebelumnya sesuai dengan n yang diinginkan. Apabila ada 5 *game* yang telah dimainkan, namun hanya ingin melihat dua *game* saja, maka akan ditampilkan 2 *game* yang paling terakhir dimainkan. Apabila n yang dimasukkan diinput lebih dari jumlah *game* yang telah dimainkan, maka semua *game* yang telah dimainkan akan ditampilkan. Jika kosong, maka hanya akan ada message yang memberitahu bahwa belum ada *game* yang telah dimainkan dan history masih kosong.

4.11 Command RESET HISTORY

Ketika command RESET HISTORY dipanggil, maka pertama akan ditampilkan pertanyaan “APAKAH KAMU YAKIN INGIN MELAKUKAN RESET HISTORY?”. Jika input yang diberikan adalah YA, maka semua daftar *game* yang telah dimainkan akan dihapus dari history dan menampilkan “HISTORY BERHASIL DI-RESET”. Jika sebaliknya, maka akan ditampilkan message “RESET HISTORY DIBATALKAN”.

4.12 Game HANGMAN

Hangman adalah sebuah game classic yang menjual permainan tebak kata. Ada dua versi permainan di dalam permainan ini (1. Hanya tersedia clue berupa digit dari kata yang diberikan dan 2. Soal terkait diberikan kepada pemain). Untuk versi yg kami gunakan disini adalah versi yang ke-2 (kami memberikan clue digit kalimat dan soal. Sebelum mulai bermain, pemain akan diminta memilih untuk langsung bermain atau mengisi kamus baru (soal dan jawaban pada file txt yang tersedia). Jika pemain memilih untuk menambah kamus baru pada soal dan jawaban yang tersedia, game akan memberikan pernyataan bahwa update file berhasil atau tidak. Selanjutnya pemain akan langsung memasuki sesi permainan utama dari game hangman ini. Game akan mengacak soal dan jawaban yang berada di dalam file txt yang tersedia. Soal pun keluar, pemain perlu menginput sebuah karakter dari ‘A’ sampai ‘Z’. bila input yang dimasukkan

lebih dari satu elemen, maka game akan meminta pemain menginput ulang sampai dengan satu elemen. Bila input huruf ada pada jawaban, maka clue baris ('_') tersebut akan berubah menjadi huruf yang ditebak. Namun, bila tebakan salah untuk ditebak atau hurufnya pernah untuk ditebak sebelumnya. Permainan akan mengeluarkan arahan kepada pemain untuk mengulang input nya sampai dengan input valid (satu karakter, karakter diantara 'A' - 'Z', dan karakter yang belum pernah untuk ditebak pada tebakan sebelumnya). Setiap permainan yang berhasil ditebak akan mengembalikan point untuk pemain (kuantitas point diberikan berdasarkan dari panjang jawaban yang disediakan).

Pemain akan dianggap kalah jika pemain telah salah dalam menebak > 10 kali. Skor yang diberikan adalah skor menang pada game terakhir kali pemain menangkan (bila pemain berakhir dengan sebuah kekalahan). Selain karena kalah, permainan juga akan dianggap selesai jika semua soal yang tersedia pada file txt telah diberikan atau ditebak. Setiap game dipastikan akan memberikan soal dan jawaban yang berbeda pada setiap stagenya. ADT yang digunakan untuk menyusun game ini adalah array, mesinkata, mesinkarakter, dan boolean. Terdapat 14 komponen (gabungan fungsi dan prosedur selain dari pada yang berada di ADT) yang menyusun. File soal dan file jawaban disimpan dan dipanggil dari soal.txt dan jawaban.txt. ASCII art generator juga turut hadir pada game ini untuk menghiasi tampilan game. Adapun untuk '#include' yang kami pakai adalah '<stdio.h>', '<time.h>', '<stdlib.h>', dan hangman.h (pada file hangman.c).

4.13 Game TOWER OF HANOI

Tower of Hanoi merupakan suatu game matematis yang melibatkan tiga buah tiang, dilambangkan dengan tiang 1, tiang 2, dan tiang 3 berurut dari kiri ke kanan. Terdapat sejumlah piringan dengan ukuran yang variatif, yang dapat dimasukkan ke tiang 1, 2, maupun 3. Saat game dimulai, pada tiang 1, akan ditampilkan 5 piringan dengan piringan paling besar di posisi paling bawah, terurut naik sehingga piringan yang paling kecil berada di posisi paling atas, sehingga membuat ilusi segitiga. Pada game ini, piringan dilambangkan dengan simbol bintang, (*) dan untuk bagian yang tidak memiliki piringan akan dilambangkan dengan garis lurus (|). Inti dari game ini adalah untuk memindahkan kelima piringan tersebut dari tiang 1 menuju tiang 3 dengan posisi yang sama dengan posisi awal, dengan catatan untuk setiap pemindahan piringan, piringan yang berada di bawahnya tidak boleh lebih kecil dibandingkan dengan piringan di atasnya.

Saat game dimulai, ditampilkan tiang 1 dengan 5 piringan yang berurut dari piringan paling besar ke piringan paling kecil dari bawah ke atas. Setelah itu, program akan meminta masukan dari pengguna, yaitu tiang asal dan tiang akhir. Jika input pengguna valid, program akan memindahkan piringan sesuai dengan permintaan pengguna. Kemudian, program akan cek validasi dari input pengguna. Input akan dinyatakan valid ketika input pengguna merupakan angka (1,2, atau 3), tiang yang dimasukan sebagai tiang asal tidak kosong, dan piringan yang ingin dipindahkan dari tiang asal lebih kecil dibandingkan piringan yang berada di tiang tujuan.

Jika input pengguna tidak valid, program akan mengeluarkan pesan bahwa input tidak valid beserta alasan input tidak valid, kemudian meminta pengguna untuk memasukan input lagi hingga input valid.

Untuk setiap pemindahan piringan, program akan menampilkan ulang ketiga tiang dengan piringan yang sudah berpindah, beserta jumlah gerakan yang sudah dilakukan oleh pengguna. Akan ditampilkan juga bahwa gerakan minimal yang dapat dilakukan oleh pengguna, atau gerakan optimal yang dapat dilakukan adalah sebanyak 31 gerakan. Game akan berhenti ketika 5 piringan yang awalnya berada di tiang 1 sudah berhasil berpindah ke tiang 3, dan kemudian akan menampilkan nilai akhir pengguna. Nilai akhir pengguna akan berdasarkan jumlah gerakan yang dilakukan. Jika gerakan yang dilakukan adalah sebanyak 31 kali, yaitu gerakan optimal, maka pengguna akan mendapatkan nilai tertinggi, yaitu 10. Jika gerakan pengguna berada di antara 50 dan 70, maka nilai yang diperoleh adalah 8. Jika gerakan yang dilakukan diantara 70 dan 90, maka nilai pengguna adalah 7. Jika gerakan yang dilakukan diantara 90 dan 100, maka nilai yang diperoleh adalah 6. Jika gerakan pengguna melebihi 100, maka nilainya akan bernilai 5. Jika pengguna mendapatkan nilai 10, program akan menampilkan pesan bahwa pengguna berhasil mendapatkan nilai tertinggi, dan untuk pengguna yang mendapatkan nilai selain 10, program akan menampilkan pesan ajakan untuk bermain kembali. ADT yang digunakan dalam penyusunan game ini adalah ADT stack, mesinkata, mesinkarakter, dan boolean. Untuk #include dalam game ini digunakan '<stdio.h>' dan towerofhanoi.h (pada file towerofhanoi.c).

4.14 Game SNAKE ON METEOR

Snake on meteor merupakan permainan yang menggunakan arena berukuran n baris x n kolom (default berukuran 5×5 namun ukuran arena dapat di setting dengan range 5 - 10) dan snake yang bergerak pada arena tersebut. Singkatnya, game ini mirip dengan game snake yang ada pada berbagai konsol lama, namun dipersulit dengan kehadiran meteor dan obstacle yang dapat di setting jumlahnya pada awal sebelum game dimulai. Terdapat beberapa fase yang dilalui player pada permainan ini. Pada awal permainan, player akan ditanyakan mengenai ukuran arena dan banyak obstacle yang ingin dimainkan. Setelah itu, akan dibuat arena, snake, dan komponen lainnya. Hal tersebut diurus menggunakan prosedur **GameSetSnake**. Setelah berhasil dibuat, game akan berjalan dengan panjang awal snake yaitu tiga, serta ukuran arena dan banyak obstacle sesuai input player.

Selanjutnya, pada fase permainan, player menginput command untuk menggerakan snake. Terdapat beberapa command yang dianggap valid, yaitu input 'A' digunakan untuk bergerak ke kiri, input 'D' digunakan untuk bergerak ke kanan, input 'W' digunakan untuk bergerak ke atas, dan input 'S' digunakan untuk bergerak ke bawah. Input selain itu akan dianggap tidak valid. Terdapat beberapa kondisi yang dianggap tidak valid, yaitu jika snake menabrak badan (diperiksa menggunakan **isNabrakBody**) dan snake menabrak meteor (diperiksa

menggunakan *isNabrakMeteor*). Selama input player tidak valid dan terjadi kondisi tidak valid, player diminta untuk melakukan input ulang dan jumlah turn tidak bertambah. Setelah input player valid, snake akan bergerak sesuai input player, di generate meteor ke arena, dan turn bertambah satu. Kondisi arena akan diprint setiap turn menggunakan prosedur *PrintArena*.

Terdapat beberapa kondisi yang dapat terjadi pada saat setelah snake bergerak, yaitu snake memakan makanan, badan snake terkena meteor, dan kondisi kalah (kondisi kepala terkena meteor, snake menabrak obstacle, atau snake sudah tidak bisa bergerak). Jika snake memakan makanan, maka panjang snake akan bertambah satu dan makanan akan degenerate ke tempat lainnya. Lalu jika badan snake terkena meteor, maka panjang snake akan berkurang satu. Lalu, jika terjadi kondisi kalah seperti kepala terkena meteor, snake menabrak obstacle, atau snake sudah tidak bisa bergerak, maka permainan akan berakhir. Permainan akan terus berlanjut sampai kondisi kalah terjadi.

Setelah permainan berakhir, program akan melakukan kalkulasi score yang didapatkan player berdasarkan panjang total snake dikali dua. Jika kepala terkena meteor, maka skor didapatkan dari panjang badan dan ekor snake dikali dua. Sedangkan untuk kasus snake menabrak obstacle dan snake sudah tidak dapat bergerak, skor didapatkan dari panjang kepala, badan, dan ekor snake dikali dengan dua.

5 Data Test

5.1 *LOAD <filename>*

Untuk Test Case pada load <filename> sebenarnya tidak berubah, namun jika load <filename> berhasil dilakukan, program juga akan memasukkan history ke dalam stack history dan memasukkan scoreboard berupa nama pemain ke dalam Set (yang nantinya dimasukkan ke list of Set dengan satu Set untuk Satu game) dan score ke dalam Map (yang nantinya dimasukkan ke list of Map dengan satu Map untuk satu game). Untuk case tersebut dapat merujuk pada gambar 5.9.1 untuk History dan 5.7.1 untuk Scoreboard hasil load file.

5.2 *CREATE GAME*

Untuk Test Case pada create game sebenarnya tidak berubah, namun jika create game berhasil dilakukan, maka program akan membentuk empty scoreboard untuk game yang berhasil di create tersebut. Untuk tampilan scoreboard setelah dilakukan create game dapat dilihat pada gambar 5.7.2

```
ENTER COMMAND: CREATE GAME
BMO: Mengenali perintah...
Masukkan nama game yang akan ditambahkan: AMOGUS
Game berhasil ditambahkan!
```

Gambar 5.2.1 Tampilan *CREATE GAME* berhasil

5.3 DELETE GAME

Untuk Test Case pada delete game sebenarnya tidak berubah, namun jika delete game berhasil dilakukan, maka program akan menghapus scoreboard untuk game tersebut dan jika user sempat memainkan game tersebut, maka program akan menghapus game dengan nama game yang sama pada history game. Untuk tampilan scoreboard setelah delete game dapat dilihat pada gambar 5.7.3

```
ENTER COMMAND: DELETE GAME
BMO: Mengenali perintah...
Berikut adalah daftar game yang tersedia
1. RRG
2. Diner DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. MARVEL SNAP
7. AMOGUS

** Hint: Game yang dapat dihapus adalah Game dengan nomor urut > 6 **
Masukkan nomor game yang akan dihapus: 7
```

Gambar 5.3.1 Tampilan DELETE GAME berhasil

5.4 PLAY GAME

Untuk Test Case pada play game berubah untuk kasus game dalam maintenance dimana kasus tersebut tidak akan terjadi kembali. Selain itu, jika play game berhasil dilakukan, maka setelah permainan selesai, program akan meminta input berupa nama player (satu kata) dan jika inputnya valid, maka akan dimasukkan ke dalam scoreboard lalu akan di sort kembali berdasarkan score dengan urutan menurun. Lalu program akan memasukkan nama game ke dalam stack of history dengan prosedur **Push**. Untuk melihat scoreboard dan history setelah melakukan play game dapat dilihat pada gambar 5.7.4 dan 5.9.2.

```
56
Ya, X adalah 56
Skor akhir Anda: 55
Masukkan Nama: ANJAY
Berhasil Menambahkan Data dalam SCOREBOARD!
```

Gambar 5.4.1 Tampilan input nama PLAY GAME berhasil

Adapun kasus jika nama player yang dimasukkan tidak valid atau sudah pernah dimasukkan sebelumnya. Jika kasus tersebut terjadi, maka program akan meminta input ulang sampai inputnya valid.

```

78
Ya, X adalah 78
Skor akhir Anda: 75
Masukkan Nama: SADIKIN
Nama Sudah Terdaftar dalam SCOREBOARD.
Harap masukkan Nama lainnya!

```

Gambar 5.4.2 Tampilan input nama PLAY GAME gagal

5.5 HELP

Terjadi penambahan list command yang ditampilkan pada command Help jika user telah melakukan konfigurasi, yang dapat dilihat pada gambar berikut.

```

ENTER COMMAND: HELP
BMO: Mengenali perintah...
DAFTAR COMMAND:
1. START - Melakukan konfigurasi sistem baru
2. LOAD <namafile.txt> - Memuat konfigurasi pada SAVEFILE yang dipilih
3. HELP - Menampilkan daftar command (perintah) yang dapat dipanggil
4. QUIT - Mengakhiri dan keluar dari sistem
5. SAVE <namafile>.txt - Melakukan penyimpanan konfigurasi pada file tertentu
6. CREATE GAME - Membuat dan menambahkan game baru ke dalam daftar
7. LIST GAME - Menampilkan daftar game yang dapat dimainkan
8. DELETE GAME - Menghapus suatu game dari dalam daftar
9. QUEUE GAME - Menambahkan game tertentu ke dalam antrian permainan
10. PLAY GAME - Memulai permainan berdasarkan antrian teratas
11. SKIP GAME <n> - Melewati n banyak game dari dalam antrian dan memainkan game berikutnya
12. SCOREBOARD - Menampilkan papan skor untuk setiap game yang terdaftar
13. RESET SCOREBOARD - Melakukan reset / penghapusan semua entries dalam papan skor
14. HISTORY <n> - Menampilkan n riwayat game yang pernah dimainkan
11. RESET HISTORY - Melakukan reset / penghapusan seluruh riwayat permainan

```

Gambar 5.5.1 Tampilan HELP setelah melakukan konfigurasi

5.6 SAVE <filename>

Untuk Test Case pada create game sebenarnya tidak berubah, namun jika SAVE <filename> berhasil dilakukan, maka selain list game, program juga akan memasukkan history dan scoreboard kedalam <filename.txt> yang dapat dilihat pada gambar berikut.

STEI- ITB	IF2111-TB2-04	Halaman 16 dari 40 halaman
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.		


```
ENTER COMMAND: SAVE save3.txt
BMO: Mengenali perintah...

-----
| Berhasil menyimpan progress. ^^ |
-----

ENTER COMMAND: 
```

```
7
RNG
Diner DASH
HANGMAN
TOWER OF HANOI
SNAKE ON METEOR
MARVEL SNAP
AVENGERS
6
AVENGERS
TOWER OF HANOI
TOWER OF HANOI
HANGMAN
TOWER OF HANOI
SNAKE ON METEOR
2
BNMO 19
FINN 12
3
JAKE 58
FINN 31
MARCELLINE 30
0
1
GA 9
1
MARSHALL 77
0
1
KADEK 41
```

Gambar 5.5.1 Tampilan SAVE setelah melakukan save dan 5.6.2 Gambar konfigurasi savefile

5.7 SCOREBOARD

Command *SCOREBOARD* akan menampilkan papan skor untuk setiap game yang terdaftar di dalam sistem. Papan skor yang ditampilkan terurut menurun berdasarkan skor, terkecuali pada game MARVEL SNAP karena sistem game PvP yang diimplementasikan. Berikut tampilan pemanggilan command *SCOREBOARD* ketika terdapat SCOREBOARD yang terisi, kosong, serta terdapat game baru yang dibuat oleh *user*.

```

ENTER COMMAND: LOAD save1.txt
BMO: Mengenali perintah...
FILE save1.txt BERHASIL DIMUAT!

ENTER COMMAND: SCOREBOARD
BMO: Mengenali perintah...
**** SCOREBOARD GAME RNG ****

| NAMA | SKOR |
|-----|-----|
| BNMO | 19 |
| FINN | 12 |
|-----|-----|

**** SCOREBOARD GAME Diner DASH ****

| NAMA | SKOR |
|-----|-----|
| JAKE | 58 |
| FINN | 31 |
| MARCELLINE | 30 |
|-----|-----|

**** SCOREBOARD GAME HANGMAN ****

| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |
|-----|-----|

**** SCOREBOARD GAME TOWER OF HANOI ****

| NAMA | SKOR |
|-----|-----|
| GA | 9 |
|-----|-----|

**** SCOREBOARD GAME SNAKE ON METEOR ****

| NAMA | SKOR |
|-----|-----|
| MARSHALL | 77 |
|-----|-----|

**** SCOREBOARD GAME MARVEL SNAP ****

| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |
|-----|-----|

ENTER COMMAND: SCOREBOARD
BMO: Mengenali perintah...
**** SCOREBOARD GAME RNG ****

| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |
|-----|-----|

**** SCOREBOARD GAME Diner DASH ****

| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |
|-----|-----|

**** SCOREBOARD GAME HANGMAN ****

| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |
|-----|-----|

**** SCOREBOARD GAME TOWER OF HANOI ****

| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |
|-----|-----|

**** SCOREBOARD GAME SNAKE ON METEOR ****

| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |
|-----|-----|

**** SCOREBOARD GAME MARVEL SNAP ****

| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |
|-----|-----|

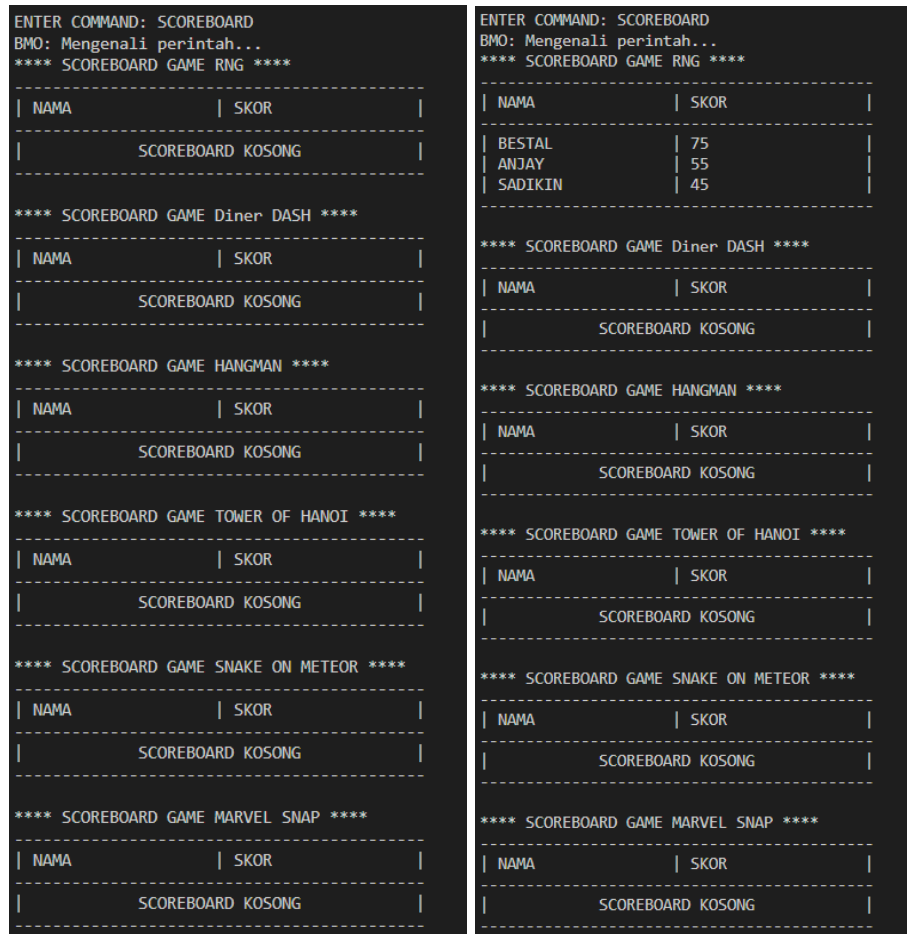
**** SCOREBOARD GAME AMOGUS ****

| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |
|-----|-----|

```

Gambar 5.7.1 Tampilan SCOREBOARD setelah LOAD dan 5.7.2: Tampilan SCOREBOARD dengan CREATE GAME

Note: Nama di dalam SCOREBOARD selalu unik dan berhuruf kapital
 Scoreboard pada 5.1.1 dan 5.1.2 merupakan hasil load dari file yang berbeda
 Scoreboard pada 5.1.2 dilakukan setelah create game

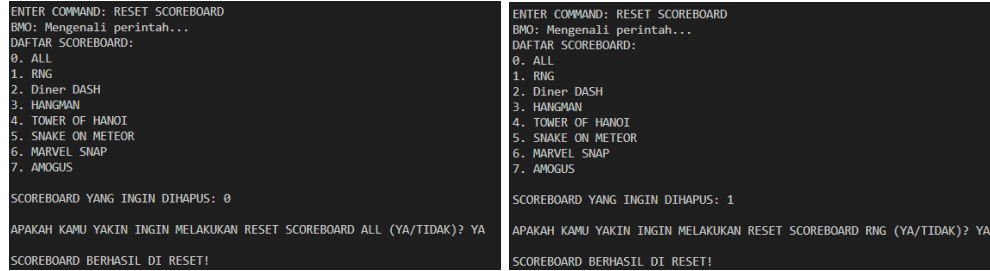


Gambar 5.7.3 Tampilan SCOREBOARD setelah DELETE GAME dan 5.7.4 Tampilan SCOREBOARD setelah PLAY GAME

5.8 RESET SCOREBOARD

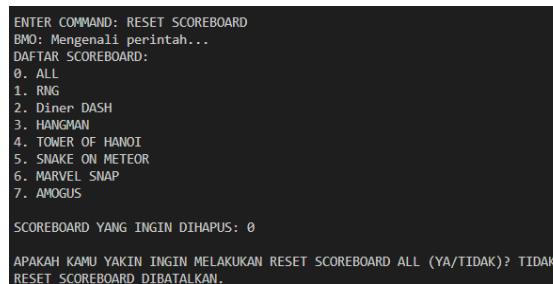
Command RESET SCOREBOARD akan melakukan reset scoreboard pada game yang dipilih oleh user melalui nomor game yang terdapat pada list game. Jika user menginput nomor game yang berada didalam range list game, maka user akan terlebih dahulu ditanyakan “APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD ALL (YA/TIDAK)?”

Jika user memilih YA, maka jika sebelumnya user menginput nomor 0 (ALL), maka program akan melakukan reset scoreboard pada seluruh game. Adapun kasus jika user menginput nomor game yang valid (kurang dari sama dengan jumlah game), maka program akan melakukan reset scoreboard pada game tersebut.



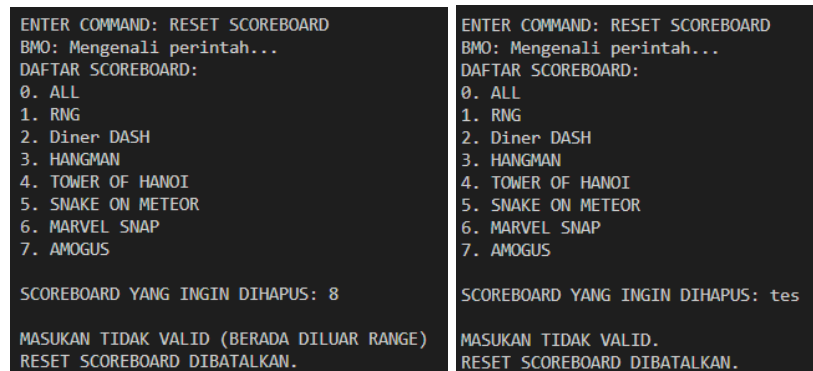
Gambar 5.8.1: **RESET SCOREBOARD** jika user memilih 0. **ALL** dan Gambar 5.8.2: jika memilih salah satu

Jika user memilih TIDAK, maka reset scoreboard akan dibatalkan dan scoreboard tidak mengalami perubahan.



Gambar 5.8.3: **RESET SCOREBOARD** jika user menjawab **TIDAK**

Adapun kasus jika nomor game yang diinput berada di luar range dan input bukan angka. Maka input akan dianggap tidak valid dan reset scoreboard dibatalkan.



Gambar 5.8.4: **RESET SCOREBOARD** jika masukan berada di luar range, dan Gambar 5.8.5: jika masukan tidak valid

5.9 HISTORY <n>

Command HISTORY <n> akan menampilkan daftar game yang telah dimainkan dengan menggunakan ADT *stack*. Command disertai dengan sebuah angka yang menentukan berapa banyak *game* (yang telah dimainkan) yang akan ditampilkan, dengan urutan pertama sebagai

game terakhir yang dimainkan. Apabila command yang dimasukkan adalah HISTORY 6, maka akan ditunjukkan enam game terakhir yang dimainkan seperti pada gambar di bawah.

Apabila command yang dimasukkan adalah HISTORY 2, maka hanya akan ditampilkan dua game terakhir yang dimainkan. Akan tetapi, bila setelah HISTORY masukan selanjutnya bukan angka, maka akan ditampilkan error “*Mohon masukkan dengan format yang benar!*”. Apabila masukan setelah HISTORY adalah angka 0 atau kurang dari 0, maka akan ditampilkan error message “*Masukan Angka Tidak Valid*”.

```
ENTER COMMAND: LOAD save1.txt
BMO: Mengenali perintah...
FILE save1.txt BERHASIL DIMUAT!

ENTER COMMAND: HISTORY 6
BMO: Mengenali perintah...

Berikut adalah daftar game yang telah dimainkan:
1. TOWER OF HANOI
2. TOWER OF HANOI
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
```

Gambar 5.9.1: *HISTORY* jika n bernilai $>$ panjang listGame

```
ENTER COMMAND: HISTORY 2
BMO: Mengenali perintah...

Berikut adalah daftar game yang telah dimainkan:
1. RNG
2. RNG
```

Gambar 5.9.2: *HISTORY* jika $0 < n \leq$ panjang listGame & Setelah playgame RNG

```
ENTER COMMAND: HISTORY GAME
BMO: Mengenali perintah...
Mohon masukkan dengan format yang benar!
** Hint: Ketik "HELP" untuk melihat perintah **
```

Gambar 5.9.3: *HISTORY* n jika masukan tidak sesuai format

```
ENTER COMMAND: HISTORY 0
BMO: Mengenali perintah...
Masukan Angka Tidak Valid.
Mohon masukkan angka yang bernilai positif!
```

Gambar 5.9.4: *HISTORY* jika n bilangan non-positif

5.10 RESET HISTORY

Command RESET HISTORY pertama-tama akan menampilkan kata-kata “APAKAH KAMU YAKIN INGIN MELAKUKAN RESET HISTORY (YA/TIDAK)?”. Dari sini, user akan diminta untuk memasukkan input lagi berupa YA atau TIDAK. Apabila dimasukkan “YA”, maka history *game* yang pernah dimainkan akan dihapus dan ditampilkan “HISTORY BERHASIL DI RESET!”. Apabila sebaliknya, maka akan ditampilkan “RESET HISTORY DIBATALKAN” dan history atau daftar *game* yang telah dimainkan.

```
ENTER COMMAND: RESET HISTORY
BMO: Mengenali perintah...
APAKAH KAMU YAKIN INGIN MELAKUKAN RESET HISTORY (YA/TIDAK)? TIDAK
RESET HISTORY DIBATALKAN.

Berikut adalah daftar game yang telah dimainkan:
1. TOWER OF HANOI
2. TOWER OF HANOI
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
```

Gambar 5.10.1: *RESET HISTORY* jika user memasukkan *TIDAK*

```
ENTER COMMAND: RESET HISTORY
BMO: Mengenali perintah...
APAKAH KAMU YAKIN INGIN MELAKUKAN RESET HISTORY (YA/TIDAK)? YA
HISTORY BERHASIL DI RESET!

ENTER COMMAND: HISTORY 5
BMO: Mengenali perintah...
Ga hanya History, Hatimu pun kosong ^^
```

Gambar 5.10.2: *RESET HISTORY* jika user memasukkan *YA*

5.11 HANGMAN

Test case pada HANGMAN akan bedakan menjadi beberapa fase, yaitu fase sebelum memulai permainan, saat melakukan permainan, dan setelah permainan selesai.

Sebelum memulai permainan, pemain akan diminta melakukan input command. Command yang dianggap valid adalah PLAYGAME, ADDWORD, dan QUIT. Jika pemain menginput command PLAYGAME, akan permainan akan dimulai. Jika pemain menginput command ADDWORD, maka user dapat menambahkan kata yang ditebak beserta soal dari kata tersebut. Jika input valid, maka kata akan dimasukkan ke dalam list jawaban dan soal akan dimasukkan ke dalam list soal. Selain itu, jika user menginput command QUIT, maka permainan akan langsung berakhir tanpa dimainkan.

```
Masukkan Command (PLAYGAME/ADDWORD/QUIT): ADDWORD
Masukkan kata berupa jawaban yang ingin dimasukkan : ANJAY
Masukan soal dari jawaban yang ingin dimasukkan: kata yang dilarang luthfi adalah...
Kata ANJAY berhasil dimasukkan ke dalam list jawaban
Soal KATA YANG DILARANG LUTHFI ADALAH... berhasil dimasukkan ke dalam list soal
```

Gambar 5.11.1 *Tampilan HANGMAN menambah kata BERHASIL*

```

Masukkan kata berupa jawaban yang ingin dimasukkan : ANJAY
Kata ANJAY sudah terdapat di dalam list jawaban
Kata ANJAY gagal dimasukkan ke dalam list jawaban

```

Gambar 5.11.2 Tampilan HANGMAN menambah kata GAGAL

```

Masukkan Command (PLAYGAME/ADDWORD/QUIT): QUIT
Anda keluar dari permainan

-----
Berhasil menyimpan listkata ^^
-----

```

Gambar 5.11.3 Tampilan HANGMAN keluar dari permainan dengan QUIT

```

Masukkan Command (PLAYGAME/ADDWORD/QUIT): PLAYGAME
Loading...
=====

KATA 1
SOAL :IKAN BERNAPAS DI AIR DENGAN?
Tebakan sebelumnya :
Kata : _ _ _ _ _
Kesempatan : 10
MASUKAN TEBAKAN :[]

```

Gambar 5.11.4 Tampilan HANGMAN memulai permainan

Jika pemain melakukan permainan, maka pemain akan diminta melakukan input berupa huruf. Jika input pemain bukan huruf, maka input dianggap tidak valid dan pemain akan diminta melakukan input kembali sampai inputnya valid. Jika input valid, maka akan dicek apakah huruf yang ditebak terdapat pada kata yang ingin ditebak. Jika input benar, maka huruf yang ditebak akan muncul pada kata yang ditebak. Namun jika input salah, kesempatan akan dikurangi satu.

```

MASUKAN TEBAKAN :A
Selamat anda berhasil menebak 1 huruf pada giliran ini
=====

KATA 1
SOAL :IKAN BERNAPAS DI AIR DENGAN?
Tebakan sebelumnya : A
Kata : _ _ _ A _ _
Kesempatan : 10
MASUKAN TEBAKAN :[]

```

Gambar 5.11.5 Tampilan HANGMAN berhasil menebak huruf

```

KATA 1
SOAL :IKAN BERNAPAS DI AIR DENGAN?
Tebakan sebelumnya : A
Kata : _ _ _ A _ _
Kesempatan : 10
MASUKAN TEBAKAN :A
Bro... Lu dah pernah nebak huruf A, coba tebak huruf lain

```

Gambar 5.11.6 Tampilan HANGMAN menebak huruf yang sudah pernah ditebak

```

MASUKAN TEBAKAN :a
Sori bro, coba situ input hurufnya di capslock dan tentu inputnya harus huruf yakk!

MASUKAN TEBAKAN :GANTENG
Input tidak valid, coba dibenerin lagi bro inputannya

```

Gambar 5.11.7 Tampilan HANGMAN input tidak valid

```

MASUKAN TEBAKAN :L
Anda salah menebak huruf, kesempatan berkurang 1
=====

                        ||
=====
KATA 1
SOAL :IKAN BERNAPAS DI AIR DENGAN?
Tebakan sebelumnya : A L
Kata : _ _ _ A _ _
Kesempatan : 9
MASUKAN TEBAKAN :[]

```

Gambar 5.11.8 Tampilan HANGMAN salah menebak huruf

Jika kata berhasil ditebak, maka user mendapatkan point sesuai dengan panjang kata yang berhasil ditebak, dan akan berlanjut ke kata berikutnya. Permainan akan terus berlanjut sama kesempatan habis atau seluruh kata pada list jawaban berhasil ditebak.

```

Selamat bro... Anda berhasil menebak kata

Jawaban benar: T E N A N G
Point anda bertambah 6
Menuju kata selanjutnya
=====

                        ||
=====
KATA 2
SOAL :SENIKMAT-NIKMATNYA MAKAN DI LUAR LEBIH NIKMAT MAKAN DI?
Tebakan sebelumnya :
Kata : _ _ _ _ _
Kesempatan : 9
MASUKAN TEBAKAN :[]

```

Gambar 5.11.9 Tampilan HANGMAN benar menebak kata


```

Kesempatan : 1
MASUKAN TEBAKAN :L
Anda salah menebak huruf, kesempatan berkurang 1
=====
TIIIIIDAAAKKK!!!!!!
AARGHHH!!      AAAAAARGHHH!!
AAAARGHHH!!    -----|| ARGHH!
AARGHHH!! O    || ARGHH!
AAAARGHHH!! /\  || ARGHH!
AARGHHH!! / \  || ARGHH!
AAAARGHHH!!    || ARGHH!
=====
Disaat kesempatan sudah habis, disitulah kepala Anda terpenggal
Anda berhasil mendapatkan 11 point

Jawaban benar: C U M A D U A A J A

```

Gambar 5.11.10 Tampilan HANGMAN kesempatan sudah habis

Jika permainan sudah selesai, maka score akhir dihitung berdasarkan total panjang dari kata yang berhasil ditebak. Artinya, jika pemain tidak bisa menebak kata satu pun, maka score akhir pemain tersebut adalah nol.

```

Tebakan sebelumnya : A Q W E R T Y U I O P
Kata : _ U _ A _ U A A _ A
Kesempatan : 1
MASUKAN TEBAKAN :L
Anda salah menebak huruf, kesempatan berkurang 1
=====
TIIIIIDAAAKKK!!!!!!
AARGHHH!!      AAAAAARGHHH!!
AAAARGHHH!!    -----|| ARGHH!
AARGHHH!! O    || ARGHH!
AAAARGHHH!! /\  || ARGHH!
AARGHHH!! / \  || ARGHH!
AAAARGHHH!!    || ARGHH!
=====
Disaat kesempatan sudah habis, disitulah kepala Anda terpenggal
Anda berhasil mendapatkan 0 point

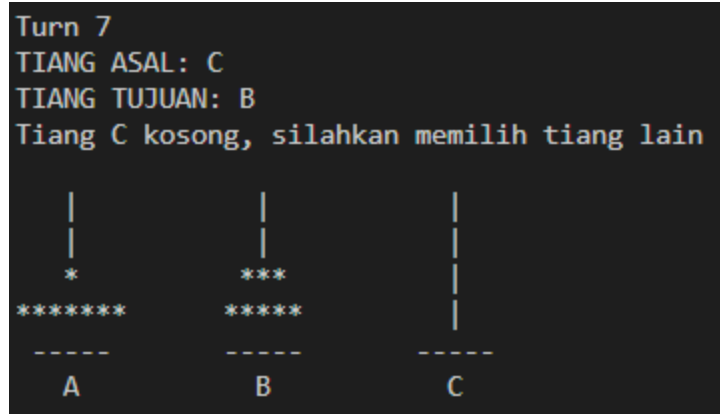
Jawaban benar: C U M A D U A A J A

```

Gambar 5.11.11 Tampilan HANGMAN kesempatan sudah habis dan skor 0

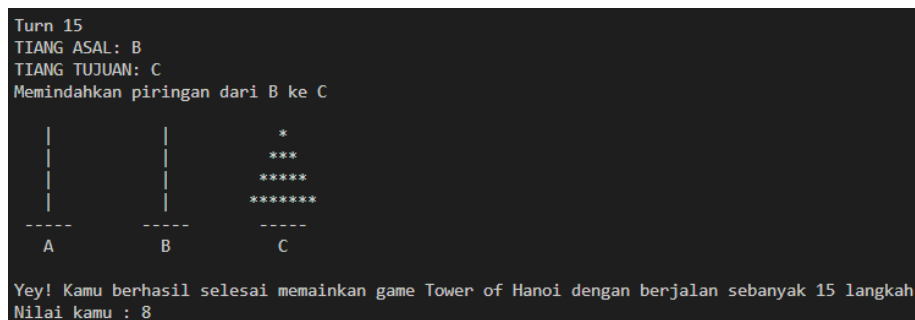
5.12 TOWER OF HANOI

Command TOWER OF HANOI akan memulai game Tower of Hanoi. Sebelum memulai permainan, user akan diminta input jumlah piringan dari tower yang ingin dimainkan. Input yang dianggap valid adalah input berupa integer lebih dari nol. Setelah memasukkan jumlah piringan, permainan akan dimulai dengan menampilkan pesan awal selamat bermain, menampilkan 3 tiang awal dengan posisi piringan yang berada di tiang 1, kemudian meminta masukan dari pengguna yang menandakan tiang asal dan tiang tujuan saat ingin memindahkan piringan. Jika input pengguna valid (tiang asal tidak kosong, input berupa angka A,B,C, piringan pada tiang asal lebih kecil dibandingkan piringan di tiang tujuan), game akan memindahkan piringan sesuai dengan input pengguna, menampilkan tiang dengan piringan yang sudah berpindah, kemudian meminta masukan pengguna lagi hingga game selesai.



Gambar 5.12.4 Tampilan TOWER OF HANOI gagal pindah piringan (tiang asal kosong)

Jika permainan sudah selesai, score akhir didapatkan dari seberapa efektif pergerakan yang dilakukan (dilihat berdasarkan jumlah turn yang dibutuhkan). Semakin sedikit turn yang dibutuhkan, semakin besar skor akhir yang didapatkan.

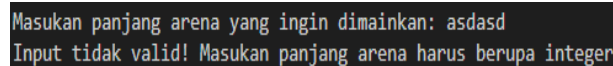


Gambar 5.12.4 Tampilan TOWER OF HANOI selesai permainan

5.13 SNAKEONMETEOR

Test case pada snake on meteor dibedakan menjadi beberapa fase, yaitu fase sebelum memulai permainan, saat melakukan permainan, dan setelah permainan selesai.

Pada awal permainan, player akan ditanyakan mengenai ukuran arena dan banyak obstacle yang ingin dimainkan. Jika input valid, akan dibuat arena, snake, dan komponen lainnya. Namun, jika input tidak valid, maka pemain akan diminta melakukan input kembali sampai input valid.



Gambar 5.13.1: Tampilan SNAKE ON METEOR sebelum permainan (input tidak valid)

```

Masukan panjang arena yang ingin dimainkan: 5
Masukan jumlah obstacle yang ingin dimainkan :2
Berikut merupakan peta permainan:
X-----X-----X-----X-----X
|       |       |       |       |
X-----X-----X-----X-----X
|       |   X   |       |       |
X-----X-----X-----X-----X
|       |       |       |       |
X-----X-----X-----X-----X
|       |   o   |   2   |   1   |   H   |
X-----X-----X-----X-----X
|       |       |   X   |       |       |
X-----X-----X-----X-----X

```

Gambar 5.13.2 Tampilan SNAKE ON METEOR sebelum permainan (input valid)

Selanjutnya, pada fase permainan, player menginput command untuk menggerakkan snake. Terdapat beberapa command yang dianggap valid, yaitu input 'A' digunakan untuk bergerak ke kiri, input 'D' digunakan untuk bergerak ke kanan, input 'W' digunakan untuk bergerak ke atas, dan input 'S' digunakan untuk bergerak ke bawah. Input selain itu akan dianggap tidak valid. Terdapat beberapa kondisi yang dianggap tidak valid, yaitu jika snake menabrak badan dan snake menabrak meteor. Selama input player tidak valid dan terjadi kondisi tidak valid, player diminta untuk melakukan input ulang dan jumlah turn tidak bertambah. Setelah input player valid, snake akan bergerak sesuai input player, melakukan generate meteor ke arena, dan turn bertambah satu. Kondisi arena akan diprint setiap turn.

```

Berikut merupakan peta permainan:
X-----X-----X-----X-----X
|       |       |       |       |
X-----X-----X-----X-----X
|       |   X   |       |       |
X-----X-----X-----X-----X
|       |       |       |       |
X-----X-----X-----X-----X
|       |   o   |   2   |   1   |   H   |
X-----X-----X-----X-----X
|       |       |   X   |       |       |
X-----X-----X-----X-----X

TURN 1
Silahkan masukkan command anda: w
Anda beruntung tidak terkena meteor! Silahkan lanjutkan permainan
Berikut merupakan peta permainan:
X-----X-----X-----X-----X
|       |       |       |       |
X-----X-----X-----X-----X
|       |   X   |       |   M   |
X-----X-----X-----X-----X
|       |       |       |   H   |
X-----X-----X-----X-----X
|       |   o   |       |   2   |   1   |
X-----X-----X-----X-----X
|       |       |   X   |       |       |
X-----X-----X-----X-----X

```

Gambar 5.13.3 Tampilan SNAKE ON METEOR berhasil melakukan pergerakan & tidak terkena meteor

```

Berikut merupakan peta permainan:
x-----x-----x-----x-----x
|       |       |       |       |
x-----x-----x-----x-----x
|       | X |       | 2 |       |
x-----x-----x-----x-----x
|       |       |       | M | 1 |
x-----x-----x-----x-----x
|       |       |       | H |       |
x-----x-----x-----x-----x
|       |       | X |       | o |
x-----x-----x-----x-----x

TURN 14
Silahkan masukkan command anda: h
Command tidak valid! Silahkan input command menggunakan huruf w/a/s/d

```

Gambar 5.13.4 Tampilan SNAKE ON METEOR gagal melakukan pergerakan (input tidak valid)

```

Berikut merupakan peta permainan:
x-----x-----x-----x-----x
|       | M |       |       |       |
x-----x-----x-----x-----x
| o | X |       |       |       |
x-----x-----x-----x-----x
|       |       |       | 3 | 2 |
x-----x-----x-----x-----x
|       |       |       | 1 |       |
x-----x-----x-----x-----x
|       |       | X |       | H |
x-----x-----x-----x-----x

TURN 15
Silahkan masukkan command anda: w
Kepala tidak dapat bergerak ke badan sendiri! silakan masukkan command lainnya

```

Gambar 5.13.5 Tampilan SNAKE ON METEOR gagal melakukan pergerakan (menabrak badan sendiri)

```

Berikut merupakan peta permainan:
x-----x-----x-----x-----x
|       | o |       |       |       |
x-----x-----x-----x-----x
| M |       |       |       |       |
x-----x-----x-----x-----x
| H | 1 | 2 | 3 | 4 |
x-----x-----x-----x-----x
|       |       |       | X |       |
x-----x-----x-----x-----x
|       |       |       |       |
x-----x-----x-----x-----x

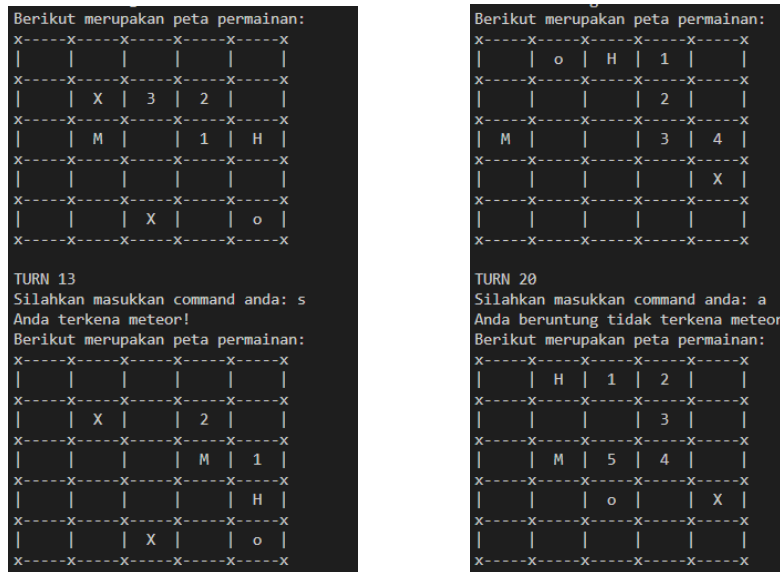
TURN 15
Silahkan masukkan command anda: w
Meteor masih panas! Anda belum dapat kembali ke titik tersebut. Silahkan masukkan command lainnya

```

Gambar 5.13.6 Tampilan SNAKE ON METEOR gagal melakukan pergerakan (menabrak meteor)

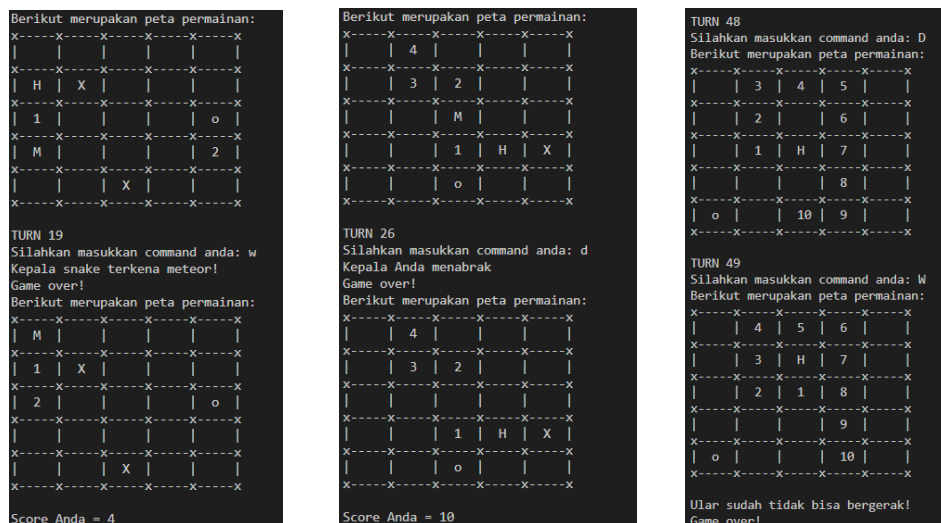
Terdapat beberapa kondisi yang dapat terjadi pada saat setelah snake bergerak, yaitu snake memakan makanan, badan snake terkena meteor, dan kondisi kalah (kondisi kepala terkena meteor, snake menabrak obstacle, atau snake sudah tidak bisa bergerak). Jika snake memakan makanan, maka panjang snake akan bertambah satu dan makanan akan degenerate ke tempat lainnya. Lalu jika badan snake terkena meteor, maka panjang snake akan berkurang satu.

Lalu, jika terjadi kondisi kalah seperti kepala terkena meteor, snake menabrak obstacle, atau snake sudah tidak bisa bergerak, maka permainan akan berakhir. Permainan akan terus berlanjut sampai kondisi kalah terjadi.



Gambar 5.13.7 Tampilan SNAKE ON METEOR badan terkena meteor dan 5.13.8 memakan makanan

Setelah permainan berakhir, program akan melakukan kalkulasi score yang didapatkan player berdasarkan panjang total snake dikali dua. Jika kepala terkena meteor, maka skor didapatkan dari panjang badan dan ekor snake dikali dua. Sedangkan untuk kasus snake menabrak obstacle dan snake sudah tidak dapat bergerak, skor didapatkan dari panjang kepala, badan, dan ekor snake dikali dengan dua.



Gambar 5.13.9 kepala terkena meteor, 5.13.10 Menabrak obstacle dan 5.13.11 Tidak dapat bergerak

6 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	LOAD <filename>	Untuk memeriksa ketika command load <filename> dipanggil, apakah program dapat meload file dengan sesuai atau tidak	LOAD <filename>	Dapat dilihat pada Data Test 5.1	Dapat dilihat pada gambar 5.7.1 dan 5.9.1	Dapat dilihat pada gambar 5.7.1 dan 5.9.1
2	CREATE GAME	Untuk memeriksa ketika command create game dipanggil, apakah program dapat membuat game sesuai dengan keinginan pengguna.	CREATE GAME Masukkan nama game yang ingin ditambahkan	Dapat dilihat pada Data Test 5.2	Dapat dilihat pada Gambar 5.2.1	Dapat dilihat pada Gambar 5.2.1
3	DELETE GAME	Untuk memeriksa ketika command delete game dipanggil, apakah program dapat menghapus game dengan sesuai.	DELETE GAME Masukkan nomor game yang ingin dihapus	Dapat dilihat pada Data Test 5.3	Dapat dilihat pada Gambar 5.3.1	Dapat dilihat pada Gambar 5.3.1
4	PLAY GAME	Untuk memainkan permainan. Game yang dipilih adalah permainan dengan urutan teratas di antrian game.	PLAY GAME	Dapat dilihat pada Data Test 5.4	Dapat dilihat pada Gambar 5.4.1, 5.4.2	Dapat dilihat pada Gambar 5.4.1, 5.4.2
5	HELP	Untuk memeriksa ketika command help dipanggil, apakah program dapat menampilkan daftar command dan penjelasannya	HELP	Dapat dilihat pada Data Test 5.5	Dapat dilihat pada Gambar 5.5.1	Dapat dilihat pada Gambar 5.5.1
6	SAVE <filename>	Untuk memeriksa ketika command save <filename> dipanggil, apakah program dapat menyimpan file sesuai dengan	SAVE <filename>	Dapat dilihat pada Data Test 5.6	Dapat dilihat pada gambar 5.6.1, 5.6.2	Dapat dilihat pada gambar 5.6.1, 5.6.2

		keinginan pengguna				
7	SCOREBOARD	Untuk memeriksa ketika command scoreboard dipanggil, apakah nilai per game yang dimainkan beserta pemainnya sudah terlihat dengan betul	SCOREBOARD	Dapat dilihat pada Data Test 5.7	Dapat dilihat pada Gambar 5.7.1, 5.7.2, 5.7.3, 5.7.4	Dapat dilihat pada Gambar 5.7.1, 5.7.2, 5.7.3, 5.7.4
8	RESET SCOREBOARD	Untuk mengetahui apakah ketika command reset scoreboard dipanggil, program dapat melakukan reset scoreboard pada scoreboard yang dipilih pengguna.	RESET SCOREBOARD Masukkan nomor scoreboard yang ingin di-reset	Dapat dilihat pada Data Test 5.8	Dapat dilihat pada gambar 5.8.1, 5.8.2, 5.8.3, 5.8.4, 5.8.5	Dapat dilihat pada gambar 5.8.1, 5.8.2, 5.8.3, 5.8.4, 5.8.5
9	HISTORY <n>	Untuk memeriksa apakah ketika command history <n> dipanggil, program dapat menampilkan n jumlah permainan yang telah dimainkan.	HISTORY <n>	Dapat dilihat pada Data Test 5.9	Dapat dilihat pada Gambar 5.9.1, 5.9.2, 5.9.3, 5.9.4	Dapat dilihat pada Gambar 5.9.1, 5.9.2, 5.9.3, 5.9.4
10	RESET HISTORY	Untuk memeriksa apakah ketika command reset history dipanggil dan setelah melakukan konfirmasi, program dapat menghapus history, dan jika tidak jadi reset program dapat menampilkan ulang history.	RESET HISTORY Masukkan input konfirmasi ("YA"/"TIDAK")	Dapat dilihat pada Data Test 5.10	Dapat dilihat pada Gambar 5.10.1, 5.10.2	Dapat dilihat pada Gambar 5.10.1, 5.10.2
11	HANGMAN	Untuk memeriksa ketika command hangman dipanggil, game hangman dapat berjalan dengan lancar.	HANGMAN	Dapat dilihat pada Data Test 5.11	Dapat dilihat pada Gambar 5.11.1, 5.11.2, 5.11.3, 5.11.4, 5.11.5, 5.11.6, 5.11.7, 5.11.8, 5.11.9, 5.11.10, 5.11.11	Dapat dilihat pada Gambar 5.11.1, 5.11.2, 5.11.3, 5.11.4, 5.11.5, 5.11.6, 5.11.7, 5.11.8, 5.11.9, 5.11.10, 5.11.11

12	TOWER OF HANOI	Untuk memeriksa ketika command tower of hanoi dipanggil, game tower of hanoi dapat berjalan dengan lancar.	TOWER OF HANOI	Dapat dilihat pada Data Test 5.12	Dapat dilihat pada Gambar 5.12.1, 5.12.2, 5.12.3, 5.12.4	Dapat dilihat pada Gambar 5.12.1, 5.12.2, 5.12.3, 5.12.4
13	SNAKE ON METEOR	Untuk memeriksa ketika command snake on meteor dipanggil, game snake on meteor dapat berjalan dengan lancar.	SNAKE ON METEOR	Dapat dilihat pada Data Test 5.13	Dapat dilihat pada Gambar 5.13.1, 5.13.2, 5.13.3, 5.13.4, 5.13.5, 5.13.5, 5.13.6, 5.13.7, 5.13.8, 5.13.9, 5.13.10, 5.13.11	Dapat dilihat pada Gambar 5.13.1, 5.13.2, 5.13.3, 5.13.4, 5.13.5, 5.13.5, 5.13.6, 5.13.7, 5.13.8, 5.13.9, 5.13.10, 5.13.11

7 Pembagian Kerja dalam Kelompok

No	Fitur/ADT/Fungsi	NIM Coder	NIM Tester
1	ADT LinkedList	18221047	18221047
2	ADT Set	18221171	18221171 18221047
3	ADT Map	18221171	18221171 18221047
4	ADT Stack	18221053	18221171 18331053 18221047
5	Main	18221171 18221047	18221171 18221047
6	Console	18221171 18221047	18221171 18221053 18221047
7	Fungsi Tower of Hanoi	18221137 18221047	18221171 18221047
8	Fungsi Hangman	18221047 18221167	18221171 18221047
9	Fungsi Snake on Meteor	18221047	18221047 18221171

NIM	Nama	Tugas
18221047	I Dewa Made Manu Pradnyana	<ol style="list-style-type: none"> 1. Mengerjakan SnakeOnMeteor. 2. Mengerjakan LinkedList. 3. Membantu mengerjakan Hangman dan Tower of Hanoi. 4. Melakukan Debugging. 5. Melengkapi laporan
18221053	Laurentia Kayleen Christopher	<ol style="list-style-type: none"> 1. Membuat HISTORY 2. Membuat RESET HISTORY 3. Membuat driver stack.h

		4. Melengkapi laporan
18221137	Felisa Aidadora Darmawan	<ol style="list-style-type: none"> 1. Mengerjakan Tower of Hanoi 2. Mengerjakan MoM asistensi 3. Melengkapi laporan
18221167	Ananda Abdul Hafizh	<ol style="list-style-type: none"> 1. Mengerjakan Hangman tanpa bonus. 2. Menyediakan referensi ADT. 3. Melengkapi laporan dan memimpin asistensi.
18221171	Hans Stephano Edbert N	<ol style="list-style-type: none"> 1. Membenarkan SAVE, LOAD, DELETEDGAME, dan CREATEGAME. 2. Membuat ADT Set dan Map dan scoreboard (dan reset scoreboard). 3. Membantu mengerjakan Hanoi 4. Menyatukan semua fungsi ke main 5. Melakukan Debugging 6. Melengkapi laporan

8 Lampiran

8.1 Deskripsi Tugas Besar 2

Buatlah sebuah permainan berbasis CLI (command-line interface). Sistem ini dibuat dalam bahasa C dengan menggunakan struktur data yang sudah kalian pelajari di mata kuliah ini. Kalian boleh menggunakan (atau memodifikasi) struktur data yang sudah kalian buat untuk praktikum pada tugas besar ini. Library yang boleh digunakan hanya stdio.h, stdlib.h, time.h dan math.h


8.2 Notulen Rapat

Form Asistensi Tugas Besar 2
IF2110/Algoritma dan Struktur Data
Sem. 1 2022/2023

No. Kelompok/Kelas : Kelompok-04/K-01
Nama Kelompok : KakCRBaikHatiDanTidakSombong
Anggota Kelompok (Nama/NIM) :
1. I Dewa Made Manu Pradnyana / 18221047
2. Laurentia Kayleen Christopher / 18221053
3. Felisa Aidadora Darmawan / 18221137
4. Ananda Abdul Hafizh / 18221167
5. Hans Stephano Edbert N / 18221171

Asisten Pembimbing : Cynthia Rusadi

Asistensi I

Tanggal : 25 November 2022	Catatan Asistensi: Kadek : ADT buat linked list itu mesti khusus untuk snake meteor gitu Kak? Aku bikinnya matrix gitu gapapa ga ya kak, jadi next column buat column selanjutnya, baris sebelum setelah, yang next prev buat snakeanya Kak Cynthia : Ini buat yang snakeanya ya? Boleh sih selagi masih implementasi linkedlist. Kadek : Buat yang obstaclenya, jumlahnya berapa ya Kak, Random, bebas atau gimana? Kak Cynthia : Di spek kalo gaada dibebasin aja Kadek : Ini aku ada beberapa test case gitu Kak, yang satu badan 1nya kena meteor, angka
Tempat : https://itb-ac-id.zoom.us/j/92725140281?pwd=Q2xPSTJlU0N2elNJRXdWY3Q1ejUvZz09	
Kehadiran Anggota Kelompok: <div style="text-align: center;">1 18221047 I Dewa Made Manu Pradnyana  2 18221053 Laurentia Kayleen Christopher</div>	

STEI- ITB	IF2111-TB2-04	Halaman 36 dari 40 halaman
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.		



3
18221137
Felisa Aidadora Darmawan



4
18221167
Ananda Abdul Hafizh



5
18221171
Hans Stephano Edbert N



belakangnya jadi 1 2, terus aku majuin jdi D, kalo yang aku tuh yang 2 nya masih ke linked list ke 1, seelumnya disini, jadi pas maju dia udah harus nyambung apa gimana?

Kak Cynthia : Ini bisa nembus? Iya bener

Kadek : Pas kena meteor, badannya mesti nyambung atau kaya gini bolong

Kak Cynthia : Iya gitu bolong, lebih mirip linkerlist sih, jadi yang blakangnya bakal ikutin depannya, yang di kamu udah bener

Apis : Kalo di hangman itu kita kan masukin tebakan huruf, kalau salah kan terpampang, kalo bener juga terpampang, misal isi huruf yang udah ada, maupun salah atau bener, itu jadinya gimana

Kak Cynthia : Itu bilang aja ga valid

Apis : Kalo dibilang jadi ga valid, diulang terus sampe suruh valid gimana Kak?

Kak Cynthia : Iya boleh gitu

Apis : Ada harapan ga Kak buat kita, atau pertanyaan balik ke kami?

Kak Cynthia : Snake on meteornya aman ga, soalnya itu emang yang paling susah

Apis : Udah dipegang oleh orang yang tepat Kak. Kalo yang bonusnya kan pake Tree, ada batasan kreativitasnya ga Kak?


Kak Cynthia : Bebas, se kreatif mungkin

Kadek : Kalo semisal H nya dipojok atas kiri sini, kan bisa tembus, kalo badannya muncul di 1 2 boleh ga Kak atau harus ikutin yang spek?





Kak Cynthia : Ikutin spek aja

Kadek : Kan di spek horizontal kebelakang, kalo udah penuh kebawah kan ya Kak? Kalo smeisalnya dibawah sini berarti memanjang ke kanan?

Kak Cynthia : Iya bener memanjang ke kanan

	Tanda Tangan Asisten: 
--	---

Asistensi II

Tanggal : 1 December 2022	Catatan Asistensi:
Tempat : https://itb-ac-id.zoom.us/j/92725140281?pwd=O2xPSTJUd0N2elNJRXdWY3O1ejUvZz09	Apis : Ketika gamenya selesai masih dirubah, ada juga perubahan di bonus kan harus ngeload file, jadi diubah struktur datanya, tapi udah 80% selesai.
Kehadiran Anggota Kelompok: <div style="text-align: center;"> <p>1</p> <p>18221047</p> <p>I Dewa Made Manu Pradnyana</p>  </div> <div style="text-align: center;"> <p>2</p> <p>18221053</p> <p>Laurentia Kayleen Christopher</p>  </div> <div style="text-align: center;"> <p>3</p> <p>18221137</p> <p>Felisa Aidadora Darmawan</p>  </div> <div style="text-align: center;"> <p>4</p> <p>18221167</p> <p>Ananda Abdul Hafizh</p>  </div>	Kayleen : history sama reset history udah selesai dan di cek di main juga udah lancar, sekarang aku mau kerjain driver buat ADT sama laporannya. Hans : Udah mulai satu-satuin semuanya, yang gaada bug itu snake on meteor, tower of hanoi sama hangman masih on process, kmrn kan aku nanya gmn nampilin marvel snap, akhirnya aku bikin nama sama scoreboard, match 1 siapa siapa, match 2 siapa siapa gapapa ga Kak? Kak Cynthia : Iya jadi masukin 2 nama, trs masukin masing-masing ke scoreboard gapapa sih Hans : Kemaren ada bug di save, tapi gatau kenapa tiba-tiba bisa. Oh iya ini ahrus di Linux ya Kak? Kak Cynthia : Iya kalo bisa harus di Linux Hans : Import filenya beda ya Kak? Soalnya smuanya malah masuk ke listgame, bukan ke tempat masing-masing, jd masih harus ada penyesuaian lagi Kak Cynthia : Kalo bisa disesuaikan sama linux, soalnya asisten rata-rata ceknya pake Linux. Minimal pake YSL gapapa. Hans : Iya tadi aku ceknya pake YSL sih Kak tp masih belom bisa, meskipun gcc nya jalan tp isinya ternyata rusak semua Kak Cynthia : OKe, itu makefilenya diperhatiin juga aja Hans : Harus bikin buat drivernya juga atau mainnya aja? Kak Cynthia : Buat mainnya aja, tapi kalo mau bikin buat drivernya juga boleh sih, siapa tau bikinnya

5
18221171
Hans Stephano Edbert N



pake banyak ADT atau gimana, tp prioritasin yang main dulu aja

Hans : Buat yang driver intinya nge run sama nunjukin hasil akhirnya ya kak?

Kak Cynthia : Iya

Hans : Progressnya lanjutin laporan, nyesuain Linux sama nyatuin yang lain yang belum

Felisa : Dari aku buat tower of hanoi kemaren masih ada yang error jadi baru selesai dibenerin, tapi masih error juga buat nge run gamenya jadi mau debugging dulu di bagian situ.

Kak Cynthia : Oke semangat deh buat tower of hanoi

Kadek : Progress aku sejauh ini udah mulai debugging snake on meteor. sejauh ini udah aman, udah nyicil laporan juga.

Kak Cynthia : Oke, mau liat snake on meteornya dong

Hans : Kalo buat testnya, nanti asistennya pake Linux ya Kak berarti?

Kak Cynthia : Kurang lebih YSL sama Linux sama. YSL itu sebenarnya Linux soalnya

Hans : Kalo laporannya, dari laporan tubes 1 diulang atau bikin baru?

Kak Cynthia : Bikin baru aja, sesuai tambahan di tubes 2


Kak Cynthia : Oh iya buat readme, tambahin juga ya cara jalanin program kalian. Kalo bisa pake makefile, tapi kalo mau pake gcc juga gapapa tpi yang penting lengkap aja jadi sisa copy

Kak Cynthia : Buat yang snake on meteor, tadi kalo 1 nya kena meteor, 2 nya bakal ikutin posisi 1 nya kan

Kadek : Iya Kak

Kak Cynthia : Oke aman kalo gitu

Hans : Yang tower of hanoi, kita tujuannya selalu pindahkan ke tiang 3 atau boleh ke 2?

	<p>Kak Cynthia : Yang penting di tiang ketiga udah full</p> <p>Hans : Oke Kak, ini bugnya disini udah dipindahin semua ke tiang 3 tapi ga ke detect end game jadi masih lanjut terus gitu</p> <p>Kak Cynthia : Ohh oke</p>
	<p>Tanda Tangan Asisten:</p> 

8.3 Log Activity Anggota Kelompok

Timeline Kelompok

No	Waktu	Keterangan
1.	Sabtu, 19 November 2022	Diskusi pembagian kerja dalam kelompok dan metode pengerjaan secara kelompok
2.	Jumat, 25 November 2022	<ul style="list-style-type: none"> - Asistensi 1 - Diskusi progress masing-masing
3.	Selasa, 29 November 2022	<ul style="list-style-type: none"> - <i>Debugging</i> code - Penyatuan program utama
4.	Kamis, 1 December 2022	<ul style="list-style-type: none"> - Asistensi 2 - Membantu kesulitan pada anggota lain, penyatuan program, penyusunan laporan
5.	Jumat, 2 December 2022	<ul style="list-style-type: none"> - Penyusunan laporan - <i>Debugging</i>