

LAPORAN TUGAS BESAR

IF2111 Algoritma dan Struktur Data

BNMO


Dipersiapkan oleh:

Kelompok 05

Rahmah Putri Azzahra	18219052
Vincent Franstyo	18221100
Karina Rahadiani	18221104
Christina Wijaya	18221106
Sultan Alta Alvaro Valencia	18221110

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		<i>IF2111-TB2-02-05</i>		<i>39</i>
		<i>Revisi</i>	<i>0</i>	<i>02/12/22</i>

Daftar Isi

1 Ringkasan	3
2 Penjelasan Tambahan Spesifikasi Tugas	4
2.1 Spesifikasi Bonus Game Hangman	4
2.2 Spesifikasi Bonus Game Tower Of Hanoi	5
2.3 Spesifikasi Bonus Game Snake On Meteor	5
3 Struktur Data (ADT)	5
3.1 ADT Stack	5
3.2 ADT Map	7
3.3 ADT ListMap	8
3.4 ADT LinkedList	8
3.5 ADT Point	9
3.6 ADT StackChar	10
4 Program Utama	11
5 Algoritma-Algoritma Menarik	15
6 Data Test	15
6.1 LOAD	15
6.2 SAVE	16
6.3 CREATE GAME	17
6.4 DELETE GAME	17
6.5 PLAY GAME	18
6.6 SKIP GAME	19
6.7 SCOREBOARD	19
6.8 RESET SCOREBOARD	20
6.9 HISTORY	21
6.10 RESET HISTORY	22
6.11 HANGMAN	23
6.12 TOWER OF HANOI	25
6.13 SNAKE ON METEOR	27
7 Test Script	28
8 Pembagian Kerja dalam Kelompok	33

9 Lampiran	34
9.1 Deskripsi Tugas Besar 2	34
Spesifikasi Umum	34
a. Command Dasar	34
b. SCOREBOARD	34
c. RESET SCOREBOARD	35
d. HISTORY <n>	35
e. RESET HISTORY	35
9.2 Notulen Rapat	37
9.3 Log Activity Anggota Kelompok	37

1 Ringkasan

BNMO adalah sebuah sistem game yang diakses melalui *command-line interface*. Program ini memiliki enam fungsi utama yaitu memainkan game, menambahkan game, menghapus game, mengurutkan game yang akan dimainkan sesuai antrian yang dimasukan, melihat history permainan, serta menampilkan scoreboard dari setiap game. Jadi user bisa menyunting game mereka juga memainkannya dalam program BNMO ini.

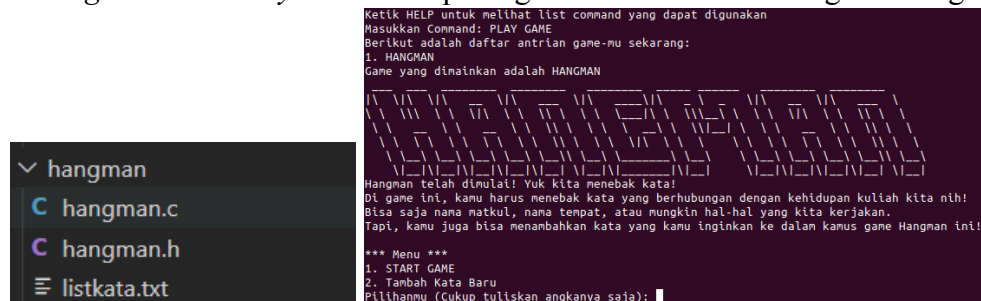
Pada pemenuhan tugas besar kali ini, pembuatan program BNMO juga diikuti oleh pemenuhan pembuatan tiga game utama tambahan, yaitu Hangman, Tower of Hanoi, dan Snake on Meteor. Hangman merupakan permainan menebak kata dengan cara memasukkan tebakan berupa huruf-huruf tertentu hingga kesempatan menebak habis. Tower of Hanoi merupakan permainan memindahkan piringan dari satu tiang ke tiang lain dengan syarat bahwa piringan yang di bawah tidak boleh lebih kecil daripada piringan yang ada di atasnya. Sedangkan dalam game Snake on Meteor, pengguna akan mengarahkan ular untuk mencari makanan sebanyak-banyaknya tanpa menabrak *obstacle* dan menguji keberuntungan agar tidak mati dihantam meteor.

Laporan ini akan memuat semua proses pembuatan sistem BNMO dimulai dari penjelasan spesifikasi, struktur data yang digunakan, program utama, data test, test script, pembagian tugas antar anggota, dan lampiran.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 Spesifikasi Bonus Game Hangman

Game Hangman pada BNMO kami menggunakan fitur tambahan, yaitu fitur penambahan *in-game dictionary* dan pembacaan dari *file*. Kami menggunakan sebuah *file* sebagai penampung kata-kata yang dapat digunakan untuk game ini, yaitu file bernama *listkata.txt*. Implementasi dari bonus *in-game dictionary* adalah dengan menyediakan pilihan bagi user untuk dapat menambahkan kata yang ingin dimasukkan ke dalam *in-game dictionary* pada awal permainan. Kata yang diinput ke dalam *in-game dictionary* ini akan dapat digunakan untuk bermain game hangman.



2.2 Spesifikasi Bonus Game Tower Of Hanoi

Pada game Tower Of Hanoi, kami mengimplementasikan spesifikasi bonus yaitu jumlah piringan yang bebas ditentukan oleh pemain dengan *range* yang berada diantara 1 hingga 30.

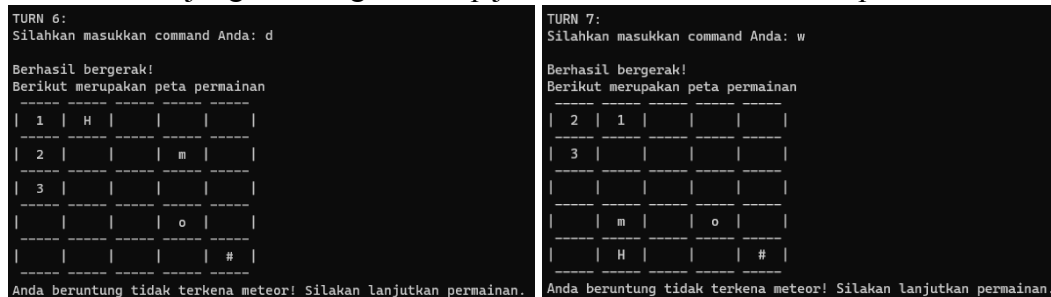
Penilaian yang digunakan adalah:

1. Jumlah minimum gerakan akan ditentukan oleh jumlah piringan yang dipilih oleh pemain dengan algoritma $(2^n - 1)$
2. Jumlah minimum gerakan yang diperlukan dan dibagi dengan 3 akan menjadi jumlah skor maksimum yang bisa diperoleh oleh pemain.



2.3 Spesifikasi Bonus Game Snake On Meteor

Pada game Snake On Meteor, kami mengimplementasikan kemunculan *obstacle* pada posisi yang tetap di dalam map 5x5 dan kepala ular yang dapat menembus dinding dan muncul di ujung seberang dari map jika telah melewati batas map.



Dapat dilihat pada gambar diatas, obstacle yang muncul akan berada di tempat yang sama dan tidak akan berpindah dimana meteor akan terus berpindah. Saat kepala ular menabrak *obstacle*, maka pemain akan kalah dan permainan akan berakhir.

Pada gambar di atas, terlihat juga bahwa jika kepala ular menembus batas map, kepala dan badan ular akan muncul di ujung seberang dari map.

3 Struktur Data (ADT)

Selain ADT yang sudah digunakan di Tugas Besar pertama, terdapat empat ADT tambahan yaitu Stack, Map, ListMap, dan LinkedList.

3.1 ADT Stack

3.1.1 Sketsa Struktur Data

```

#define NilStack -1
#define MaxElStack 100

typedef int infotypeStack;
typedef int address;

typedef struct
{
    infotypeStack T[MaxElStack];
    address TOPSTACK;
} Stack;

#define TopStack(S) (S).TOPSTACK
#define InfoTopStack(S) (S).T[(S).TOPSTACK]

```

Berikut primitif yang digunakan dalam ADT ini:

1. Konstruktor / Kreator:
 - a. CreateEmptyStack (Stack *S)
2. Predikat Test Keadaan Stack:
 - a. IsEmptyStack (Stack S)
 - b. IsFullStack (Stack S)
3. Pop/Push:
 - a. PopStack (Stack *S, infotypeStack X)
 - b. PushStack (Stack *S, infotypeStack *X)
4. Prototype:
 - a. PrintStack (Stack S)
 - b. InverseStack (Stack S)

3.1.2 Persoalan yang dapat diselesaikan

ADT Stack ini digunakan untuk game Tower Of Hanoi. Pada game Tower Of Hanoi, digunakan ADT Stack of Integer untuk penanda jumlah piringan yang ada pada tiang. Jumlah piringan tersebut akan digunakan untuk membuat tiang yang akan ditampilkan.

3.1.3 Alasan Pemilihan ADT

ADT stack ini digunakan untuk memindahkan piringan dari yang paling atas ke piringan lainnya dan berurut sampai piringan terbawah. ADT Stack cocok untuk digunakan pada kasus ini karena ADT ini memiliki dua buah primitif yang memenuhi kriteria tersebut, yaitu Pop (memindahkan piringan teratas) dan Push (memasukkan piringan ke paling atas).

3.1.4 Implementasi ADT

driver_stack.c, stack.c, stack.h

3.2 ADT Map

3.2.1 Sketsa Struktur Data

```
#define NilMap 0
#define MaxElMap 100
#define Undefined -999

typedef char *keytype;
typedef int valuetype;
typedef int address;

typedef struct
{
    keytype Key;
    valuetype Value;
} infotype;

typedef struct
{
    infotype Elements[MaxElMap];
    address CountMap;
} Map;
```

Berikut primitif yang digunakan dalam ADT ini:

1. Konstruktor/Kreator :
 - a. void CreateEmpty(Map *M)
2. Predikat:
 - a. boolean IsEmpty(Map M)
 - b. boolean IsFull(Map M)
3. Operator:
 - a. valuetype Value(Map M, keytype k)
 - b. void Insert(Map *M, keytype k, valuetype v)
 - c. void Delete(Map *M, keytype k)
 - d. boolean IsMember(Map M, keytype k)

3.2.2 Persoalan yang dapat diselesaikan

ADT Map digunakan untuk scoreboard. Map berguna sebagai *database* sederhana yang menyimpan hanya dua elemen yang terdiri dari *key* dan *value*. *key* akan diisi data berupa nama *user* yang harus unik namanya karena akan menjadi petunjuk untuk *value* yang berisi *score* dari *user* setelah memainkan *game*. *user* yang terlebih dahulu memasukan nama mereka kemudian disusul *user* lain yang memiliki *score* yang sama akan tetap dicatat lebih awal.

3.2.3 Alasan Pemilihan ADT

ADT ini digunakan karena diperlukannya *database* yang sederhana hanya berupa dua data yaitu nama dan *score*. Map menjadi ADT yang tepat karena sesuai definisinya, Map memiliki dua elemen yaitu *key* dan *value* yang selalu berpasangan, dimana *key* akan selalu memetakan *value*.

3.2.4 Implementasi sebagai ADT

map.h, map.c, driver_map.c, listMap.h, listMap.c, dan driver_listMap.c

3.3 ADT ListMap

3.3.1 Sketsa Struktur Data

```
typedef int IdxTypeMap;

typedef struct
{
    ElTypeMap peta[10];
    int Neff;
} ListMap;
```

Berikut primitif yang digunakan dalam ADT ini:

1. Konstruktor/Kreator :
 - a. void createListMap(ListMap *L)
2. Operasi-operasi :
 - a. void insertToList(ListMap *L, Map M)
 - b. void resetAllMap(ListMap *L)
 - c. void resetAtMap(ListMap *L, int n)
 - d. deleteListMapAt(ListMap *L, int n)
 - e. printAllList(ListMap L)

3.3.2 Persoalan yang dapat diselesaikan

ADT ListMap digunakan untuk scoreboard. ListMap berguna untuk menampilkan *scoreboard* dari masing-masing game. Dimana setiap *game* memiliki *scoreboard* masing-masing yang memiliki *user* dan *score* yang berbeda-beda pula. Fungsi utama dari ADT ini guna mengubah (menghapus atau menambahkan) *scoreboard* sesuai keberadaan *game*.

3.3.3 Alasan Pemilihan ADT

ADT ini digunakan karena *scoreboard* yang bertipe Map tidak hanya satu melainkan banyak. Akan lebih baik jika Map-Map ini dijadikan List of Map guna mempermudah operasi-operasi penghapusan atau penambahan pada *scoreboard*.

3.3.4 Implementasi sebagai ADT

listMap.h, listMap.c, dan driver_listMap.c

3.4 ADT LinkedList

3.4.1. Sketsa Struktur Data

```
typedef char* infotypeListdp;
typedef struct tElmtlistdp *addressListdp;
typedef struct tElmtlistdp {
    infotypeListdp info;
```



```

        addressListdp next;
        addressListdp prev;
        POINT pos;
    } ElmtListdp;
    typedef struct {
        addressListdp First;
        addressListdp Last;
    } Listdp;

```

Berikut primitif yang digunakan dalam ADT ini:

1. Konstruktor/Kreator :
 - a. void CreateEmptyListdp(Listdp *L)
2. Operasi-operasi :
 - a. addressListdp SearchListdp(Listdp L, POINT Y)
 - b. void InsVLastListdp(Listdp *L, infotypeListdp X, POINT Y)
 - c. void DelFirstListdp(Listdp *L, addressListdp *P)
 - d. void DelLastListdp(Listdp *L, addressListdp *P)
 - e. void DelPListdp(Listdp *L, POINT X)
 - f. POINT GetLastPosListdp(Listdp L)
3. Print:
 - a. void PrintForwardListdp(Listdp L)
 - b. void PrintBackwardListdp(Listdp L)
4. Alokasi Dealokasi :
 - a. addressListdp AlokasiListdp(infotypeListdp X, POINT Y)
 - b. void DealokasiListdp(addressListdp P)
5. Tes Empty :
 - a. boolean IsEmptyListdp(Listdp L)

3.4.2. Persoalan yang dapat diselesaikan

ADT LinkedList digunakan pada game Snake On Meteor. ADT ini digunakan saat membuat badan pada snake tersebut. Fungsi utama ADT ini adalah untuk insert dan delete pada badan snake tersebut.

3.4.3. Alasan Pemilihan ADT

ADT ini digunakan karena badan pada snake di Snake On Meteor saling berhubungan. Game tersebut juga membutuhkan fungsi fungsi di ADT ini seperti insert dan delete pada badan di snake tersebut.

3.4.4. Implementasi sebagai ADT

listdp.h, listdp.c, dan driver_listdp.c

3.5 ADT Point

3.5.1. Sketsa Struktur Data

```
typedef struct {
    int x; /* absis */
    int y; /* ordinat */
} POINT;
```

Berikut primitif yang digunakan dalam ADT ini:

1. Konstruktor/Kreator :
 - a. POINT MakePOINT (int X, int Y)
2. Input :
 - a. void BacaPOINT (POINT * P)
3. Cek Equal:
 - a. boolean EQ (POINT P1, POINT P2)
 - b. boolean NEQ (POINT P1, POINT P2)
4. Cek Lokasi :
 - a. boolean IsOrigin (POINT P)
 - b. boolean IsOnSbX (POINT P)
 - c. boolean IsOnSbY (POINT P)
 - d. int Kuadran (POINT P)

3.5.2. Persoalan yang dapat diselesaikan

ADT Point ini digunakan di game Snake On Meteor. ADT ini digunakan untuk menentukan posisi dari badan snake, meteor, food, dan obstacle pada board map 5 x 5.

3.5.3. Alasan Pemilihan ADT

ADT ini digunakan karena kita perlu menentukan letak setiap objek pada game snake on meteor(badan, meteor, obstacle, dan food). Sehingga dengan ADT point ini kita bisa mempermudah untuk menentukan letak objek - objek tersebut pada board game.

3.5.4. Implementasi sebagai ADT

point.h, point.c, driver_point.c

3.6 ADT StackChar

3.6.1. Sketsa Struktur Data

```
#define NilStackchar -1
#define MaxElStackchar 100

typedef char *infotypeStackchar;
typedef int address;

typedef struct
{
    infotypeStackchar T[MaxElStackchar];
    address TOPSTACKCHAR;
```

```
} Stackchar;
```

Berikut primitif yang digunakan dalam ADT ini:

1. Konstruktor/Kreator :
 - a. void CreateEmptyStackChar(Stackchar *S)
2. Cek Empty/Full :
 - a. boolean IsEmptyStackChar(Stackchar S)
 - b. boolean IsFullStackChar(Stackchar S)
3. Operasi Pop/Push:
 - a. void PushStackChar(Stackchar *S, infotypeStackchar X)
 - b. void PopStackChar(Stackchar *S, infotypeStackchar *X)
4. Operasi operasi lain :
 - a. void PrintStackChar(Stackchar S
 - b. void CopyStackChar (Stackchar sIn, Stackchar * sOut)
 - c. void InverseStackChar (Stackchar *S)
 - d. Stackchar MergeStackChar (Stackchar s1, Stackchar s2)

3.6.2. Persoalan yang dapat diselesaikan

ADT StackChar ini digunakan untuk menyimpan history pada program. Fungsi Push digunakan untuk menambahkan history saat setelah Play Game. Fungsi Pop digunakan ketika mereset History.

3.6.3. Alasan Pemilihan ADT

ADT ini digunakan karena History pada program menggunakan konsep stack yaitu setelah memainkan game, game tersebut akan disimpan di history paling atas. Jadi History membutuhkan fungsi Push yang ada di ADT StackChar ini.

3.6.4. Implementasi sebagai ADT

stackchar.h, stackchar.c,

4 Program Utama

Berikut adalah algoritma utama yaitu main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#include "../src/game_util/game_util.h"
#include "../src/games/diner_dash/diner_dash.h"
#include "../src/games/random_number_generator/random_number_generator.h"
#include "../src/games/game_tambahan/game_tambahan.h"
#include "../src/games/hangman/hangman.h"
#include "../src/games/TicTacToe/tictactoe.h"
#include "../src/main_util/main_util.h"
#include "../src/games/SnakeOnMeteor/snakeonmeteor.h"
#include "../src/games/TowerOfHanoi/TowerOfHanoi.h"
```



```

}
while (cek)
{
    printf("\nKetik HELP untuk melihat list command yang dapat digunakan\n");
    printf("Masukkan Command: ");
    STARTINPUTKATA();
    if (IsEqual(currentWord, "CREATE GAME"))
    {
        createGame(&ListGame, &Scoreboard, &Temp);
    }
    else if (IsEqual(currentWord, "LIST GAME"))
    {
        listGame(&ListGame);
    }
    else if (IsEqual(currentWord, "DELETE GAME"))
    {
        deleteGame(&ListGame, QueueGame, &Scoreboard, &History);
    }
    else if (IsEqual(currentWord, "QUEUE GAME"))
    {
        queueGame(&QueueGame, ListGame);
    }
    else if (IsEqual(currentWord, "PLAY GAME"))
    {
        playGame(ListGame, &QueueGame, &History, &Scoreboard);
    }
    else if ((IsEqual(takeword(currentWord, 1), "SKIP")) &&
(IsEqual(takeword(currentWord, 2), "GAME")))
    {
        if (takeword(currentWord, 3).Length == 0)
        {
            printf("Tolong masukkan nomor game yang ingin di skip dengan format
'SKIP GAME <n>'!\n");
        }
        else {
            skipGame(WordToInt(takeword(currentWord, 3)), ListGame, &QueueGame,
&History, &Scoreboard);
        }
    }
    else if (IsEqual(currentWord, "HELP"))
    {
        help();
    }
    else if (IsEqual(takeword(currentWord, 1), "SAVE"))
    {
        if (currentWord.Length == 4)
        {
            printf("Tolong masukkan nama file!\n");
        }
        else
        {
            Word x = takeword(currentWord, 2);
            save(WordToString(x), ListGame, History, Scoreboard);
        }
    }
    else if (IsEqual(currentWord, "SCOREBOARD"))
    {
        scoreboard(Scoreboard, ListGame);
    }
    else if (IsEqual(currentWord, "RESET SCOREBOARD"))
    {
        resetScoreboard(&Scoreboard, &ListGame);
    }
    else if (IsEqual(takeword(currentWord, 1), "HISTORY"))
    {
        if (takeword(currentWord, 2).Length == 0)

```

```

        {
            printf("Tolong masukkan jumlah history game yang ingin ditampilkan
dengan format 'HISTORY <n>'!\n");
        }
        else{
            history(WordToInt(takeword(currentWord, 2)), History);
        }
    }
    else if (IsEqual(currentWord, "RESET HISTORY"))
    {
        resetHistory(&History);
    }
    else if (IsEqual(currentWord, "QUIT"))
    {
        quit(ListGame, History, Scoreboard);
    }
    else
    {
        otherCommand();
    }
}
return 0;
}

```

Program utama dimulai dengan pendefinisian header yang boleh digunakan untuk spesifikasi yaitu <stdio.h>, <stdlib.h>, <time.h>, <math.h> dan semua header dari semua fungsi/prosedur maupun ADT yang telah dibuat. Terdapat enam header yang telah dibuat yaitu game_util.h, diner_dash.h, random_number_generator.h, game_tambahan.h, hangman.h, tictactoe.h, main_util.h, snakeonmeteor.h, dan TowerOfHanoi.h.

Setelah pendefinisian semua header yang dibutuhkan, dilanjutkan oleh pembuatan fungsi displaybinomo() yang berfungsi sebagai header selamat datang saat baru memulai program.

Kemudian masuk ke bagian utama dari program int main() yang di dalamnya diawali dengan pemanggilan fungsi displaybinomo(). Sebelum memanggil fungsi/prosedur tertentu, diperlukan adanya deklarasi parameter yaitu ListGame berbentuk ArrayOfGame yang dipakai untuk MakeArrayOfGame, QueueGame berbentuk Queue yang dipakai untuk CreateQueue, ListMap berbentuk List of Map untuk Scoreboard, dan pendefinisian boolean cek. Validasi ini kemudian dipakai untuk menentukan looping menuju program utama.

Setelah memenuhi syarat looping, akan dicetak sebuah perintah untuk memasukan command untuk START atau LOAD guna menjalankan program utama. Pembacaan command dari user kemudian dilakukan dengan ADT mesin kata yang dilakukan sesuai dengan arahan spesifikasi. Jika user memasukan command LOAD maka harus diikuti nama file yang kemudian akan dibaca sistem. Jika memasukan command START, user akan langsung bisa mengakses semua command yang terdapat di BNMO yaitu CREATE GAME, LIST GAME, DELETE GAME, QUEUE GAME, PLAY GAME, SKIP GAME, HISTORY, RESET HISTORY, SCOREBOARD, RESET SCOREBOARD, HELP, SAVE, dan QUIT. Dari masing-masing command yang berhasil dibaca, akan langsung memanggil masing-masing fungsi.

5 Algoritma-Algoritma Menarik

Tidak ada algoritma yang menarik.

6 Data Test

Berikut data-data test yang terdapat dalam program.

6.1 LOAD

Program akan membaca file yang disimpan oleh sistem yang berisi game-game yang akan dimasukkan ke list game, history permainan sebelumnya yang akan dimasukkan ke stack history, serta scoreboard dari masing-masing game yang tersimpan sebelumnya yang akan dimasukkan ke list map scoreboard. Program akan membaca 2 parameter, yaitu kata “LOAD” dan nama file yang akan dibaca. Jika user hanya menginput satu kata “LOAD” tanpa menginput nama file akan keluar pesan “Tolong masukkan nama file”. Jika file yang diinput user tidak ditemukan maka akan keluar pesan “Masukkan file dengan benar”. Jika berhasil maka keluar pesan “Save file berhasil dibaca. BNMO berhasil dijalankan”.

```
/$$$$$$ \ /$$$$$ / $ $ \ $ $ /$$$$$ $ $ \ /$ $ /$$$$$ \
$ $ | _$ $ $ $ | $$$ \ $ $ $ $ | $$$ \ /$$$ $ $ | $ $
$ $ $$$< $ $ | $$$ $ $ $ $ | $ $ $$$ /$$$ $ $ | $ $
$ $ | _$ $ | _$ $ | $$$ $$$ \ $ $ $$$ / $ $ $ $ \ $ $
$ $ $ $ / $ $ | $ $ | $$$ $ $ $ $ / $ $ | $ / $ $ $ $ $ $ /
$$$$$$ / $$$$$$ / $ $ / $ $ / $$$$$$ / $ $ / $ $ / $$$$$$ /

Masukkan Command (START / LOAD <file_name>): LOAD
Tolong masukkan nama file!
Masukkan Command (START / LOAD <file_name>):
```

Gambar 6.1.1 Tampilan ketika hanya input “LOAD” tanpa nama file

```
/$$$$$$ \ /$$$$$ / $ $ \ $ $ /$$$$$ $ $ \ /$ $ /$$$$$ \
$ $ | _$ $ $ $ | $$$ \ $ $ $ $ | $$$ \ /$$$ $ $ | $ $
$ $ $$$< $ $ | $$$ $ $ $ $ | $ $ $$$ /$$$ $ $ | $ $
$ $ | _$ $ | _$ $ | $$$ $$$ \ $ $ $$$ / $ $ $ $ \ $ $
$ $ $ $ / $ $ | $ $ | $$$ $ $ $ $ / $ $ | $ / $ $ $ $ $ $ /
$$$$$$ / $$$$$$ / $ $ / $ $ / $$$$$$ / $ $ / $ $ / $$$$$$ /

Masukkan Command (START / LOAD <file_name>): LOAD KamuNanyea?
Masukkan file dengan benar
Masukkan Command (START / LOAD <file_name>):
```

Gambar 6.1.2. Tampilan ketika input nama file yang tidak ditemukan di data

```
/$$$$$$ \ /$$$$$ / $ $ \ $ $ /$$$$$ $ $ \ /$ $ /$$$$$ \
$ $ | _$ $ $ $ | $$$ \ $ $ $ $ | $$$ \ /$$$ $ $ | $ $
$ $ $$$< $ $ | $$$ $ $ $ $ | $ $ $$$ /$$$ $ $ | $ $
$ $ | _$ $ | _$ $ | $$$ $$$ \ $ $ $$$ / $ $ $ $ \ $ $
$ $ $ $ / $ $ | $ $ | $$$ $ $ $ $ / $ $ | $ / $ $ $ $ $ $ /
$$$$$$ / $$$$$$ / $ $ / $ $ / $$$$$$ / $ $ / $ $ / $$$$$$ /

Masukkan Command (START / LOAD <file_name>): LOAD savefile.txt
Save file berhasil dibaca. BNMO berhasil dijalankan.
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: _
```

Gambar 6.1.3 Tampilan ketika berhasil

6.2 SAVE

Program akan menyimpan state game pemain saat ini ke dalam suatu file. Program akan membaca 2 parameter, yaitu kata “SAVE” dan nama file yang akan disimpan pada sistem. Jika user hanya menginput satu kata “SAVE” tanpa menginput nama file akan keluar pesan “Tolong masukkan nama file!”. Jika nama

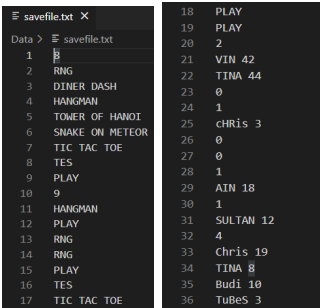
file yang diinput belum ada pada folder, maka program akan otomatis membuat file dengan nama tersebut pada folder Data. Jika nama file yang diinput sudah ada pada folder, maka program akan melakukan overwrite terhadap file tersebut. Hal-hal yang disimpan yaitu list game, history, dan scoreboard masing-masing game. Setelah berhasil maka program akan mengeluarkan pesan “Save file berhasil disimpan”.

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: SAVE
Tolong masukkan nama file!
```

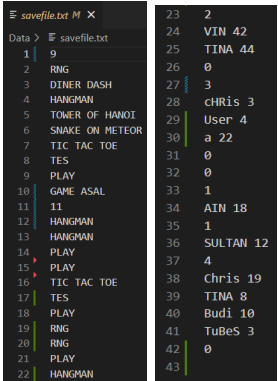
Gambar 6.2.1 Tampilan ketika hanya input “SAVE” tanpa nama file

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: SAVE savefile.txt
Save file berhasil disimpan.
```

Gambar 6.2.2 Tampilan ketika berhasil



Gambar 6.2.3 Tampilan awal file sebelum di-save



Gambar 6.2.4 Tampilan akhir file setelah di-save

6.3 CREATE GAME

Ketika prosedur CREATE GAME dipanggil, program akan meminta input nama game baru yang akan dimasukkan ke list game. Jika nama game yang diinput sudah ada di list game maka akan muncul pesan “Nama game tersebut sudah ada dalam list. Game gagal ditambahkan.”. Jika nama game yang diinput belum ada, game tersebut akan ditambahkan ke list game sekaligus dibuatkan map kosong ke

list of map untuk scoreboard game tersebut, serta program akan mengeluarkan pesan “Game berhasil ditambahkan”.

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: CREATE GAME
Masukkan nama game yang akan ditambahkan: RNG
Nama game tersebut sudah ada dalam list. Game gagal ditambahkan.
```

Gambar 6.3.1 Tampilan saat game yang dibuat sudah ada di list game

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: CREATE GAME
Masukkan nama game yang akan ditambahkan: GAME ASAL
Game berhasil ditambahkan
```

Gambar 6.3.2 Tampilan ketika berhasil

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: LIST GAME
Berikut adalah daftar game yang tersedia
1. RNG
2. DINER DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. TIC TAC TOE
7. TES
8. PLAY
```

Gambar 6.3.3 Tampilan awal list game sebelum game baru dibuat

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: LIST GAME
Berikut adalah daftar game yang tersedia
1. RNG
2. DINER DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. TIC TAC TOE
7. TES
8. PLAY
9. GAME ASAL
```

Gambar 6.3.4 Tampilan akhir list game setelah game baru dibuat

6.4 DELETE GAME

Ketika prosedur DELETE GAME dipanggil, program akan meminta input berupa integer yang merupakan indeks game yang ingin dihapus. Jika indeks yang diinput merupakan bilangan antara 1 sampai 6 maka game tidak berhasil dihapus dan mengeluarkan pesan “Game tidak dapat dihapus”. Jika indeks kurang dari 1 maka input tidak valid dan mengeluarkan pesan “Maaf, nomor yang Anda masukkan tidak valid.”. Jika indeks melebihi angka di list game maka input tidak valid dan mengeluarkan pesan “Maaf, nomor yang Anda masukkan melebihi jumlah game yang ada.”. Jika indeks lebih dari 6 dan kurang dari banyaknya game maka input valid dan game berhasil dihapus sekaligus map di scoreboard untuk game tersebut dihapus dan mengeluarkan pesan “Game berhasil dihapus”.

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: DELETE GAME
Berikut adalah daftar game yang tersedia
1. RNG
2. DINER DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. TIC TAC TOE
7. TES
8. PLAY
9. GAME ASAL
Masukkan nomor yang akan dihapus :-9
Maaf, nomor yang Anda masukkan tidak valid.
```

Gambar 6.4.1 Tampilan ketika nomor game tidak valid (kurang dari 1)

```

Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: DELETE GAME
Berikut adalah daftar game yang tersedia
1. RING
2. DINER DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. TIC TAC TOE
7. TES
8. PLAY
9. GAME ASAL
Masukkan nomor yang akan dihapus :10
Maaf, nomor yang Anda masukkan melebihi jumlah game yang ada.

```

Gambar 6.4.2 Tampilan ketika nomor game melebihi jumlah game yang ada

```

Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: DELETE GAME
Berikut adalah daftar game yang tersedia
1. RING
2. DINER DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. TIC TAC TOE
7. TES
8. PLAY
9. GAME ASAL
Masukkan nomor yang akan dihapus :1
Game tidak dapat dihapus

```

Gambar 6.4.3 Tampilan ketika nomor game yang di-input termasuk game default

```

Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: DELETE GAME
Berikut adalah daftar game yang tersedia
1. RING
2. DINER DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. TIC TAC TOE
7. TES
8. PLAY
9. GAME ASAL
Masukkan nomor yang akan dihapus :9
Game berhasil dihapus

```

Gambar 6.4.4 Tampilan ketika berhasil

6.5 PLAY GAME

Ketika prosedur PLAY GAME dipanggil, program akan menampilkan daftar antrian game yang dimiliki pengguna. Jika tidak ada game dalam daftar antrian game pengguna, maka program akan menampilkan “Maaf, antrian game-mu kosong. Silakan menambahkan game ke antrian terlebih dahulu dengan mengetik command: QUEUE GAME”. Jika terdapat game dalam daftar antrian game pengguna, program akan mulai memainkan game pada urutan pertama. Di akhir permainan, program akan memasukkan nama game yang baru saja dimainkan ke history. Selanjutnya, program akan menampilkan skor akhir pengguna, lalu meminta inputan nama pengguna untuk dimasukkan ke scoreboard. Jika nama yang dimasukkan sudah ada pada scoreboard, maka program akan menampilkan “Nama pengguna tersebut sudah ada di scoreboard. Silakan input nama yang berbeda!”. Jika belum ada, maka program akan menyimpan nama dan score tersebut pada scoreboard dan menampilkan pesan “Namamu berhasil ditambahkan ke scoreboard!”.

```

Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: PLAY GAME
Berikut adalah daftar antrian game-mu sekarang:
Maaf, antrian game-mu kosong. Silakan menambahkan game ke antrian terlebih dahulu dengan
mengetik command: QUEUE GAME

```

Gambar 6.5.1 Tampilan ketika daftar antrian game kosong

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: PLAY GAME
Berikut adalah daftar antrian game-mu sekarang:
1. HANGMAN
Game yang dimainkan adalah HANGMAN
```

Gambar 6.5.2 Tampilan ketika daftar antrian game tidak kosong

```
--- GAME OVER ---
Skor akhir: 22
Nama: CHRIS
Nama pengguna tersebut sudah ada di scoreboard. Silakan input nama yang berbeda!
Nama: cHRis
Nama pengguna tersebut sudah ada di scoreboard. Silakan input nama yang berbeda!
Nama: chris
Nama pengguna tersebut sudah ada di scoreboard. Silakan input nama yang berbeda!
Nama:
```

Gambar 6.5.3 Tampilan ketika permainan berakhir dan user memasukkan nama yang sudah ada pada scoreboard

```
--- GAME OVER ---
Skor akhir: 4
Nama: User
Namamu berhasil ditambahkan ke scoreboard!
```

Gambar 6.5.4 Tampilan ketika berhasil memasukkan nama pada scoreboard

```
**** SCOREBOARD HANGMAN ****
| NAMA | SKOR |
|-----|-----|
| cHRis | 3    |
| User  | 4    |
```

Gambar 6.6.4 Tampilan akhir scoreboard

6.6 SKIP GAME

Pada dasarnya sama seperti Tugas Besar 1. Yang berbeda hanya pada saat memainkan game, karena pada akhir game harus diminta inputan nama pemain untuk dimasukkan ke scoreboard. Tampilan akhir tersebut sama persis seperti tampilan akhir pada prosedur PLAY GAME, sehingga tidak diberikan contoh gambarnya lagi pada data test ini.

6.7 SCOREBOARD

Ketika prosedur Scoreboard dipanggil, program akan menampilkan semua tabel scoreboard dari masing-masing game. Jika scoreboard masih kosong, maka program akan menampilkan pesan “Scoreboard Kosong” ke layar. Jika tidak, maka program akan menampilkan seluruh nama dan skor dari pengguna di masing-masing game yang dimainkan.


```

SCOREBOARD YANG INGIN DIHAPUS :3
APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD HANGMAN
(YA/TIDAK)?YA
Scoreboard berhasil di-reset.

Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: SCOREBOARD
**** SCOREBOARD RNG ****
| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |

**** SCOREBOARD DINER DASH ****
| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |

**** SCOREBOARD HANGMAN ****
| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |

```

Gambar 6.8.3 Reset scoreboard 3

```

SCOREBOARD YANG INGIN DIHAPUS :8
APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD ALL
(YA/TIDAK)?YA
Scoreboard berhasil di-reset.

Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: SCOREBOARD
**** SCOREBOARD RNG ****
| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |

**** SCOREBOARD DINER DASH ****
| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |

**** SCOREBOARD HANGMAN ****
| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |

**** SCOREBOARD TOWER OF HANOI ****
| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |

**** SCOREBOARD SNAKE ON METEOR ****
| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |

**** SCOREBOARD TIC TAC TOE ****
| NAMA | SKOR |
|-----|-----|
| SCOREBOARD KOSONG |

```

Gambar 6.8.4 Reset scoreboard ALL

```

Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: RESET SCOREBOARD
DAFTAR SCOREBOARD :
0. ALL
1. RNG
2. DINER DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. TIC TAC TOE
SCOREBOARD YANG INGIN DIHAPUS :8
Mohon masukkan jawaban yang benar.

```

Gambar 6.8.5 Reset Scoreboard yang melebihi indeks

```

Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: RESET SCOREBOARD
DAFTAR SCOREBOARD :
0. ALL
1. RNG
2. DINER DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. TIC TAC TOE
SCOREBOARD YANG INGIN DIHAPUS :0
APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD ALL
(YA/TIDAK)?TIDAK
Scoreboard tidak jadi di-reset.

```

Gambar 6.8.6 Tampilan saat tidak jadi reset scoreboard

6.9 HISTORY

Ketika prosedur HISTORY dipanggil, user diminta memasukkan 2 parameter dengan format HISTORY <n>. Jika input valid maka game yang terakhir dimainkan akan ditampilkan. Jika user hanya menginput history tanpa diikuti

sebuah integer maka program akan mengeluarkan pesan “Tolong masukkan jumlah history game yang ingin ditampilkan dengan format “History <n>!””.

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: HISTORY
Tolong masukkan jumlah history game yang ingin ditampilkan dengan format 'HISTORY <n>!'
```

Gambar 6.9.1 History tanpa jumlah

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: HISTORY 2
Berikut adalah daftar Game yang telah dimainkan
1. TOWER OF HANOI
2. DINER DASH
```

Gambar 6.9.2 Tidak semua history

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: HISTORY 6
Berikut adalah daftar Game yang telah dimainkan
1. TOWER OF HANOI
2. DINER DASH
3. HANGMAN
4. TIC TAC TOE
5. RNG
6. RNG
```

Gambar 6.9.3 History ditampilkan semua

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: HISTORY 10
-----History Kosong-----
```

Gambar 6.9.4 History kosong

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: HISTORY !)
Mohon masukkan angka yang benar, yaitu lebih dari 0.
```

Gambar 6.9.5 Jumlah history tidak benar

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: HISTORY 6
Berikut adalah daftar Game yang telah dimainkan
1. HANGMAN
2. TIC TAC TOE
3. RNG
4. RNG
```

Gambar 6.9.6 Tampilan saat jumlah history melebihi jumlah sebenarnya

6.10 RESET HISTORY

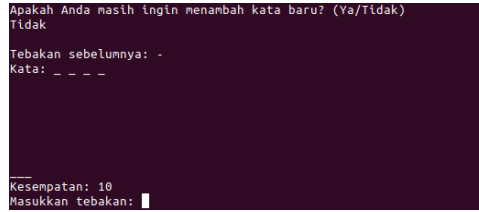
Command Reset History digunakan untuk menghilangkan history yang telah tercatat sebelumnya. Dengan memanggil command ini dan mengkonfirmasi reset, maka history yang ada akan dihapus dan tidak ada history permainan lagi. Tetapi, dengan memasukkan “TIDAK”, history tidak akan dihapus dan akan kembali

```
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: RESET HISTORY
APAKAH KAMU YAKIN INGIN MELAKUKAN RESET HISTORY? (YA/TIDAK) Y
History berhasil di-reset.

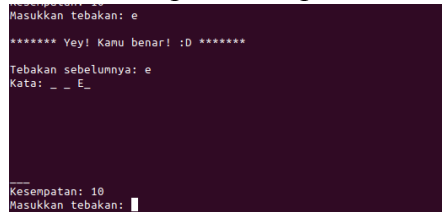
Ketik HELP untuk melihat list command yang dapat digunakan
Masukkan Command: HISTORY !)
Mohon masukkan angka yang benar, yaitu lebih dari 0.
```

Gambar 6.10.1 Reset History

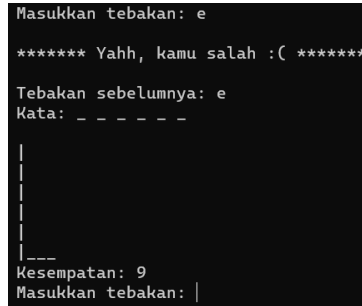
Gambar 6.11.3 Tampilan saat kata baru sudah ada dalam list kata



Gambar 6.11.4 Tampilan saat permainan dimulai



Gambar 6.11.5 Tampilan saat tebakan sesuai



Gambar 6.11.6 Tampilan saat tebakan tidak sesuai



Gambar 6.11.7 Tampilan saat seluruh tebakan telah terisi


```

Tebakan sebelumnya: -
Kata: - - - -

Kesenjangan: 1
Masukkan tebakan: q
***** Yahn, kamu salah :( *****

Kesenjangan: 0
Kata yang benar adalah BNMO
Kesenjangan menebakmu telah habis! Permainan berakhir...
Kamu memperoleh total poin sebesar: 20 poin!

--- GAME OVER ---
Skor akhir: 20
Nama:

```

Gambar 6.11.8 Tampilan saat kesempatan tebakan habis

```

Masukkan kata baru: ahahah1
Maaf, kata yang dimasukkan tidak boleh mengandung karakter selain huruf.
Apakah Anda masih ingin menambah kata baru? (Ya/Tidak)

```

Gambar 6.11.9 Tampilan saat ada karakter non-huruf dalam kata

```

Tebakan sebelumnya: a
Kata: - - - -

Kesenjangan: 9
Masukkan tebakan: a
***** Huruf sudah ditebak. *****

```

Gambar 6.11.10 Tampilan saat huruf yang sudah ditebak ditebak lagi

6.12 TOWER OF HANOI

Tower of Hanoi adalah permainan yang memindahkan disc dari tiang A menuju tiang C dengan ketentuan-ketentuan yang berlaku, yaitu

1. Tidak dapat memindahkan disc ke tiang yang sama dengan tiang semulanya
2. Tidak dapat memindahkan disc ke tiang dengan disc yang lebih kecil daripada disc yang ingin dipindahkan

Scoring dari Tower of Hanoi ini bergantung pada minimal move yang perlu dijalankan. Minimal move sendiri didapatkan dengan rumus $2^n - 1$. Saat jumlah pergerakan seseorang sama dengan minimal pergerakan yang dibutuhkan, pemain akan mendapatkan score maksimum yang bisa didapatkan.

```

=====
Tower of Hanoi
=====
Hanoi game yang akan dimainkan ke urutan: 4
Kata berhasil ditambahkan ke urutan
Sebelum ini adalah list game mu
1. Tower of Hanoi

Ketika Hanoi untuk melihat list command yang dapat digunakan
Masukkan command: Hanoi game
Berhasil untuk melihat list command game mu sekarang:
1. Tower of Hanoi
Game yang dimainkan adalah Tower of Hanoi

=====
Tower of Hanoi
=====
Selamat datang di Tower of Hanoi!
Masukkan jumlah disk yang ingin dimainkan: 5
Anda memiliki 5 disk

Untuk memainkan permainan ini, silakan ikuti instruksi berikut:
1. Pastikan nama tiang awal dari disk yang ingin dipindahkan (A/B/C) pada bagian "Tiang Asal"
2. Nama tiang awal dan tiang tujuan tidak boleh sama
3. Pastikan nama tiang tujuan dari disk yang ingin dipindahkan (A/B/C) pada bagian "Tiang Tujuan"
4. Disk yang akan dipindahkan ke tiang yang memiliki disk teratas yang lebih besar dari disk yang dipindahkan
5. Ketika "Exit" pada "Tiang Tujuan" akan keluar dari permainan atau saat anda telah menyerah
Setelah disk yang dipindahkan akan dilihat ulang kembali 1 langkah
6. Akan disediakan minimal move yang perlu dilakukan untuk menyelesaikan permainan ini
Setiap langkah yang diambil akan dihitung move atau pergerakan. Jika anda menyerah 1
7. Minimal Move yang ada bergantung pada jumlah disk yang dimainkan
8. Jika anda menyerah, maka skor yang didapat adalah 0
9. Jika anda menyelesaikan permainan, maka skor yang didapat adalah 1/3 dari langkah minimal yang dibutuhkan

Your current score: 33
The minimum move you have to make: 31

=====
Tiang A:
=====

```

Gambar 6.12.1 Tampilan Awal Game Tower Of Hanoi

```

Tiang Asal: A
Tiang Tujuan: C
Memindahkan piringan ke C...

TURN 2

  |      |      |
  ***    |      |
  *****|      |
  *****|      |
  *****|      |
  -----|      |
  A      B      C

Tiang Asal: █

```

Gambar 6.12.2 Pemindahan Disc

```

Tiang Asal: A
Tiang Tujuan: C
Memindahkan piringan ke C...

Disc pada Tower tujuan tidak boleh lebih kecil dari Tower awal
TURN 2

  |      |      |
  ***    |      |
  *****|      |
  *****|      |
  *****|      |
  -----|      |
  A      B      C

Tiang Asal: █

```

Gambar 6.12.3 Tampilan saat disc di bawah lebih kecil daripada disc di atas

```

TURN 10

  |      |      |
  |      |      |
  *      |      |
  *****|      |
  -----|      |
  A      B      C

Tiang Asal: C
Tiang Tujuan: C
Tiang Asal dan Tiang Tujuan tidak boleh sama!

```

Gambar 6.12.4 Tampilan saat tiang awal dan akhir sama

```

TURN 7

  |      |      |
  |      |      |
  *      |      |
  -----|      |
  A      B      C

Tiang Asal: d
Tiang Tujuan: A
Tiang Asal tidak valid!
Tiang Asal: A
Tiang Tujuan: d
Tiang Tujuan tidak valid!

```

Gambar 6.12.5 Tampilan saat tiang tujuan awal atau tujuan akhir tidak valid

```

Anda memiliki sisa 10 langkah
TURN 32

  |      |      |
  |      |      |
  |      |      |
  |      |      |
  *      |      |
  -----|      |
  A      B      C

Tiang Asal: A
Tiang Tujuan: C
Memindahkan piringan ke C...

Anda memiliki sisa 9 langkah
Selamat! Anda berhasil menyelesaikan permainan ini!
Score kamu adalah 9
Skor akhir: 9
Nama: █

```

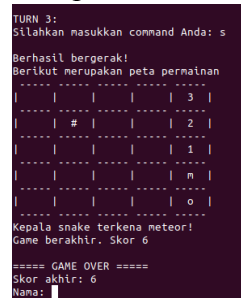
Gambar 6.12.6 Tampilan saat game berakhir

6.13 SNAKE ON METEOR

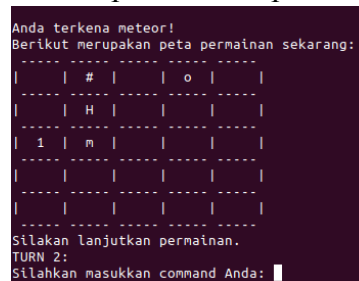
Snake on meteor adalah permainan yang menggerakkan seekor ular untuk mendapatkan makanan agar ular dapat bertambah panjang. Semakin panjang ular, maka score yang didapatkan juga akan semakin banyak. Tetapi, pada permainan ini, terdapat penghalang yang perlu dihindari dan meteor yang akan muncul secara random. Saat kepala ular menabrak penghalang atau tertimpa meteor, permainan berakhir dan score yang didapatkan adalah sisa jumlah tubuh dikali dengan 2.



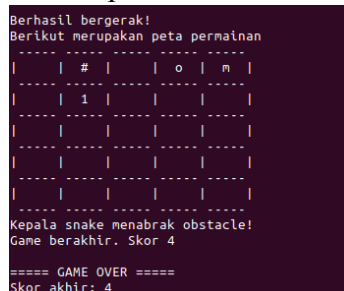
Gambar 6.13.1 Tampilan Awal Snake On Meteor



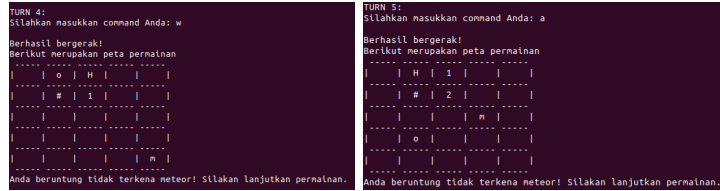
Gambar 6.13.2 Tampilan saat kepala terkena meteor



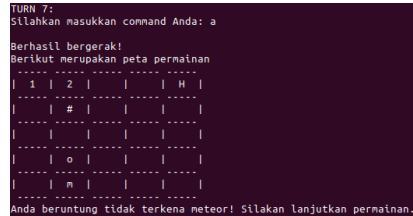
Gambar 6.13.3 Tampilan saat tubuh terkena meteor



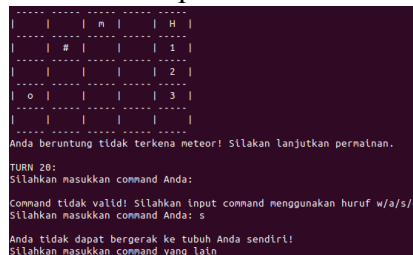
Gambar 6.13.4 Tampilan saat tubuh menabrak obstacle



Gmabar 6.13.5 Tampilan saat menerima makanan



Gambar 6.13.6 Tampilan saat menembus map



Gambar 6.13.7 Tampilan saat badan menabrak diri sendiri

7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	LOAD	Memastikan prosedur LOAD gagal jika input tanpa nama file	Memasukkan command LOAD tanpa nama file	Gambar 6.1.1	Gambar 6.1.3	Gambar 6.1.1
2	LOAD	Memastikan prosedur LOAD gagal jika input salah	Memasukkan command LOAD dengan nama file yang salah	Gambar 6.1.2	Gambr 6.1.3	Gambar 6.1.2
3	LOAD	Memastikan prosedur LOAD berjalan dengan baik ketika input benar	Memasukkan command LOAD dengan nama file yang sesuai	Gambar 6.1.3	Gambar 6.1.3	Gambar 6.1.3
4	SAVE	Memastikan prosedur SAVE gagal	Memasukkan command SAVE	Gambar 6.2.1	Gambar 6.2.2	Gambar 6.2.1

		berjalan ketika input salah	tanpa diikuti nama file			
5	SAVE	Memastikan prosedur SAVE berjalan dengan baik ketika input sesuai	Memasukkan command SAVE dengan diikuti input nama file	Gambar 6.2.2	Gambar 6.2.2, Gambar 6.2.4	Gambar 6.2.2
6	CREATE GAME	Memastikan game masuk ke list game dan map scoreboard masuk ke listofmap jika input sesuai	Memasukkan command CREATE GAME dan memberi input nama game yang belum ada di list game	Gambar 6.3.2	Gambar 6.3.2	Gambar 6.3.2
7	CREATE GAME	Memastikan game tidak masuk ke list game jika nama game yang diinput sudah ada di list game	Memasukkan command CREATE GAME dan memberi input nama game yang sudah ada di list game	Gambar 6.3.1	Gambar 6.3.2	Gambar 6.3.1
8	DELETE GAME	Memastikan prosedur gagal jika game yang dihapus game default	Memasukan command DELETE GAME lalu menghapus game default	Gambar 6.4.3	Gambar 6.4.4	Gambar 6.4.3
9	DELETE GAME	Memastikan prosedur gagal jika nomor game yang dimasukan tidak valid	Memasukan command DELETE GAME lalu menghapus game dengan urutan yang tidak valid	Gambar 6.4.1, 6.4.2	Gambar 6.4.4	Gambar 6.4.1, 6.4.2
10	DELETE GAME	Memastikan prosedur bisa menghapus game	Memasukan command DELETE GAME saat program dijalankan dan memberi input urutan yang valid	Gambar 6.4.4	Gambar 6.4.4	Gambar 6.4.4
11	PLAY GAME	Memastikan prosedur bisa memainkan game dan berhasil memasukkan nama user yang valid ke scoreboard	Memasukan command PLAY GAME saat program dijalankan lalu memberi input user yang belum ada di scoreboard	Gambar 6.5.2, 6.5.4	Gambar 6.5.2, 6.5.4	Gambar 6.5.2, 6.5.4
12	PLAY GAME	Memastikan prosedur bisa memainkan	Memasukan command PLAY GAME saat	Gambar 6.5.3	Gambar 6.5.3	Gambar 6.5.3

		game dan berhasil memasukkan nama user yang tidak valid ke scoreboard	program dijalankan lalu memberi input user yang sudah ada di scoreboard			
13	PLAY GAME	Memastikan prosedur tidak berjalan saat tidak ada antrian game	Memasukan command PLAY GAME saat game tidak ada di antrian game	Gambar 6.5.1	Gambar 6.5.1	Gambar 6.5.1
14	SKIP GAME	Memastikan prosedur bisa melewati antrian game	Memasukkan command SKIP GAME diikuti input an integer yang valid	-	-	-
15	SKIP GAME	Memastikan prosedur gagal saat input lebih jumlah dari antrian	Memasukkan command SKIP GAME diikuti input integer yang melebihi jumlah game di antrian game	-	-	-
16	SKIP GAME	Memastikan prosedur gagal saat tidak ada antrian game	Memasukkan command SKIP GAME ketika antrian game sedang kosong	-	-	-
17	SCOREBOARD	Memastikan prosedur scoreboard berhasil ditampilkan	Memasukkan command SCOREBOARD	Gambar 6.7.1	Gambar 6.7.1	Gambar 6.7.1
18	RESET SCOREBOARD	Memastikan prosedur berjalan dan scoreboard berhasil direset	Memasukkan command RESET SCOREBOARD dan diikuti dengan input indeks yang valid lalu input "YA"	Gambar 6.8.1, 6.8.2, 6.8.3, 6.8.4	Gambar 6.8.1, 6.8.2, 6.8.3, 6.8.4	Gambar 6.8.1, 6.8.2, 6.8.3, 6.8.4
19	RESET SCOREBOARD	Memastikan prosedur tidak jadi dijalankan jika user menginput "TIDAK"	memasukkan command RESET SCOREBOARD dan diikuti dengan input indeks yang valid lalu input "TIDAK"	Gambar 6.8.6	Gambar 6.8.6	Gambar 6.8.6
20	RESET SCOREBOARD	Memastikan prosedur tidak dijalankan jika input dari user tidak valid	Memasukkan command RESET SCOREBOARD dan diikuti inout	Gambar 6.8.5	Gambar 6.8.5	Gambar 6.8.5

			indeks yang tidak valid			
21	HISTORY	Memastikan prosedur tidak dijalankan ketika user tidak menginput indeks	Memasukkan command HISTORY tanpa diikuti input sebuah integer	Gambar 6.9.1	Gambar 6.9.1	Gambar 6.9.1
22	HISTORY	Memastikan prosedur dijalankan ketika user menginput indeks yang valid	Memasukkan command HISTORY diikuti input sebuah indeks yang valid	Gambar 6.9.2, 6.9.3, 6.9.4	Gambar 6.9.2, 6.9.3, 6.9.4	Gambar 6.9.2, 6.9.3, 6.9.4
23	HISTORY	Memastikan prosedur tidak dijalankan ketika input user tidak valid	Memasukkan command HISTORY diikuti input sebuah indeks yang tidak valid (Bukan integer)	Gambar 6.9.5	Gambar 6.9.5	Gambar 6.9.5
24	RESET HISTORY	Memastikan prosedur dijalankan ketika user menginput "YA"	Memasukkan command RESET HISTORY dengan input setelahnya adalah "YA"	Gambar 6.10.1	Gambar 6.10.1	Gambar 6.10.1
25	RESET HISTORY	Memastikan prosedur tidak jadi dijalankan ketika user menginput "TIDAK"	Memasukkan command RESET HISTORY dengan input setelahnya adalah "TIDAK"	Gambar 6.10.2	Gambar 6.10.2	Gambar 6.10.2
26	HANGMAN	Memastikan HANGMAN bisa dimainkan dengan baik	Menambahkan HANGMAN ke antrian game dan masukkan command PLAY GAME	Gambar 6.11.1	Gambar 6.11.1	Gambar 6.11.1
27	HANGMAN	Memastikan bisa menambahkan kata baru yang valid ke game	Memainkan HANGMAN lalu memasukkan input "2" lalu memasukkan input kata baru yang belum ada di HANGMAN	Gambar 6.11.2	Gambar 6.11.2	Gambar 6.11.2
28	HANGMAN	Memastikan tidak bisa menambahkan kata baru yang	Memainkan HANGMAN lalu memasukkan input "2" lalu memasukkan input	Gambar 6.11.3	Gambar 6.11.3	Gambar 6.11.3

		sudah ada di game	kata baru yang sudah ada di HANGMAN			
29	HANGMAN	Memastikan game HANGMAN saat menang	Memainkan HANGMAN menang	Gambar 6.11.7	Gambar 6.11.7	Gambar 6.11.7
30	HANGMAN	Memastikan game HANGMAN saat kalah	Memainkan HANGMAN lalu kalah	Gambar 6.11.8	Gambar 6.11.8	Gambar 6.11.8
31	HANGMAN	Memastikan tidak bisa menambahkan kata baru yang tidak valid ke game	Memainkan HANGMAN lalu memasukkan input “2” lalu memasukkan input kata baru yang tidak valid ke HANGMAN	Gambar 6.11.9	Gambar 6.11.9	Gambar 6.11.9
32	TOWER OF HANOI	Memastikan TOWER OF HANOI bisa dimainkan dengan baik	Menambahkan TOWER OF HANOI ke antrian game dan masukkan command PLAY GAME	Gambar 6.12.1	Gambar 6.12.1	Gambar 6.12.1
33	TOWER OF HANOI	Memastikan game tidak berjalan ketika pemindahan disc tidak sesuai aturan	Memainkan TOWER OF HANOI lalu memindahkan disc yang lebih besar ke tower yang ada disc yang lebih kecil	Gambar 6.12.3	Gambar 6.12.3	Gambar 6.12.3
34	TOWER OF HANOI	Memastikan game tidak berjalan ketika input tower asal atau tower tujuan tidak valid atau tidak sesuai aturan	Memainkan TOWER OF HANOI lalu memberi input an tower asal atau tujuan yang tidak sesuai aturan	Gambar 6.12.4, 6.12.5	Gambar 6.12.4, 6.12.5	Gambar 6.12.4, 6.12.5
35	TOWER OF HANOI	Memastikan game TOWER OF HANOI saat menang	Memainkan TOWER OF HANOI lalu memenangkan	Gambar 6.12.6	Gambar 6.12.6	Gambar 6.12.6
36	SNAKE ON METEOR	Memastikan SNAKE ON METEOR bisa dimainkan dengan baik	Menambahkan SNAKE ON METEOR ke antrian game lalu memasukkan command PLAY GAME	Gambar 6.13.1	Gambar 6.13.1	Gambar 6.13.1

37	SNAKE ON METEOR	Memastikan game over ketika terjadi kondisi kalah	Memainkan SNAKE ON METEOR lalu head terkena meteor atau head menabrak obstacle	Gambar 6.13.2, 6.13.4	Gambar 6.13.2, 6.13.4	Gambar 6.13.2, 6.13.4
38	SNAKE ON METEOR	Memastikan badan bertambah ketika snake memakan makanan	Memainkan SNAKE ON METEOR lalu mengarahkan head ke food	Gambar 6.13.5	Gambar 6.13.5	Gambar 6.13.5
39	SNAKE ON METEOR	Memastikan meminta input an ulang ketika head diarahkan ke badan sendiri	Memainkan SNAKE ON METEOR lalu memberi input an head ke arah badannya sendiri	Gambar 6.13.7	Gambar 6.13.7	Gambar 6.13.7
40	SNAKE ON METEOR	Memastikan badan berkurang 1 ketika meteor mengenai badan snake	Memainkan SNAKE ON METEOR lalu meteor mengenai badan snake	Gambar 6.13.3	Gambar 6.13.3	Gambar 6.13.3
41	SNAKE ON METEOR	Memastikan snake bisa menembus tembok pada map	Memainkan SNAKE ON METEOR lalu mengarahkan head ke arah tembok pada map	Gambar 6.13.6	Gambar 6.13.6	Gambar 6.13.6

8 Pembagian Kerja dalam Kelompok

Nama - NIM	Pembagian Tugas
Rahmah Putri Azzahra - 18219052	
Vincent Franstyo - 18221100	ADT Stack + Driver, Tower of Hanoi, Laporan
Karina Rahadiani - 18221104	Scoreboard + Reset Scoreboard, ADT Map + Driver, ADT ListMap + Driver, Laporan
Christina Wijaya - 18221106	History + Reset History, Snake on Meteor, Hangman, Laporan
Sultan Alta Alvaro Valencia - 18221110	Snake on Meteor, ADT LinkedList + Driver, Laporan

9 Lampiran

9.1 Deskripsi Tugas Besar 2

Spesifikasi Umum

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**

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 tugas besar 1.

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 tugas besar 1.
2. Diner Dash
Dijelaskan pada tugas besar 1.
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 menebak huruf yang tidak terdapat dalam kata.
- Pada setiap giliran, pemain menebak satu huruf yang terdapat pada kata tersebut. Apabila huruf tebak 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 kecil daripada piringan yang ada di atasnya. BNMO ingin membuat permainan ini dan agar lebih mengerti mengenai permainan ini. 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*)
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.

9.2 Notulen Rapat

Rapat Pertama	
Tanggal	19/11/2022
Meeting untuk membahas tubes secara keseluruhan, penentuan tanggal asistensi, dan pembagian tugas Pembagian tugas awal <ol style="list-style-type: none"> 1. Scoreboard + Reset, ADT Map + Driver → Karina 2. History + Reset, Snake on Meteor → Christina 3. Snake on Meteor, ADT LinkedList + Driver → Sultan 4. ADT Stack + Driver, Tower of Hanoi → Vincent 5. Game utk Tree, ADT Tree+ Driver → Rahma Asistensi : 24/11/22	

Rapat Kedua	
Tanggal	29/11/2022
Meeting untuk merevisi tugas besar bersama termasuk debug error bersama Asistensi : 01/12/22	

9.3 Log Activity Anggota Kelompok

1. Rahmah Putri Azzahra/18219052
 24/11/2022 : Asistensi 1
2. Vincent Franstyo/18221100
 - 21/11/2022 : Menambah ADT Stack
 - 23/11/2022 : Visual Game Tower Of Hanoi
 - 24/11/2022 : Mekanisme Tower Of Hanoi selesai, memperbaiki visual
 - 24/11/2022 : Menyelesaikan visual Game Tower Of Hanoi
 - 24/11/2022 : Memperbaiki fungsi” ter-redefined dalam stack
 - 24/11/2022 : Asistensi 1
 - 26/11/2022 : Memperbaiki redefined function stack
 - 26/11/2022 : Menambah instruksi untuk Tower Of Hanoi

29/11/2022 : Update readme
 29/11/2022 : Update redefined in ADTs
 1/12/2022 : Update readme
 1/12/2022 : Update scoring in Tower Of Hanoi
 2/12/2022 : Update arraydin dan Hangman

3. Karina Rahadiani/18221104

22/11/2022 : Menambah ADT Map
 22/11/2022 : Update Prosedur skip game
 23/11/2022 : Membuat prosedur scoreboard
 23/11/2022 : Membuat ADT List of Map
 24/11/2022 : Update Scoreboard
 24/11/2022 : Asistensi 1
 25/11/2022 : Memperbaiki InsertMap
 27/11/2022 : Memperbaiki prosedur scoreboard dan menambah reset scoreboard
 28/11/2022 : Update ADT Map dan ListMap beserta driver
 29/11/2022 : Update Help dan Update delete game
 29/11/2022 : Menambah driver map
 29/11/2022 : Offline Bug Fixing
 29/11/2022 : Update redefined in ADTs
 1/12/2022 : Update driver map

4. Christina Wijaya/18221106

24/11/2022 : Update game Hangman
 24/11/2022 : Finalisasi Game Hangman + Bonus
 24/11/2022 : Update Play Game & Skip Game
 24/11/2022 : Membuat fungsi history dan resethistory
 24/11/2022 : Membuat ADT Stack Char
 24/11/2022 : Asistensi 1
 26/11/2022 : Memperbaiki fungsi play game, save, skip game, ADT Map, List, dkk
 28/11/2022 : Finalize Snake On Meteor
 29/11/2022 : Update main function
 29/11/2022 : Offline Bug Fixing
 29/11/2022 : Update redefined in ADTs
 29/11/2022 : Update reset history, history, save, hangman, snakeonmeteor
 30/11/2022 : Update play game, skip game, delete game
 30/11/2022 : Fixed SNAKE on Meteor & map
 2/12/2022 : Update snake on meteor
 2/12/2022 : Update path in Hangman

5. Sultan Alta Alvaro Valencia/18221110

20/11/2022 : Menambah ADT Linked List

21/11/2022 : Menambah ADT Point dan Update ADT Linked List
 21/11/2022 : Menambah Snakeonmeteor.c dan Snakeonmeteor.h
 21/11/2022 : Update Prosedur Load
 22/11/2022 : Update Prosedur Load
 24/11/2022 : Asistensi 1
 26/11/2022 : Memperbaiki load dan main function
 29/11/2022 : Update RNG and TTT
 29/11/2022 : Offline Bug Fixing
 29/11/2022 : Update redefined in ADTs
 1/12/2022 : Fixed Snake on meteor
 1/12/2022 : Update linkedList Driver
 2/12/2022 : Update driver dan prosedur Load