LAPORAN TUGAS BESAR

IF2111 Algoritma dan Struktur Data

BNMO

Dipersiapkan oleh:

Kelompok 5:

Richard Haris	18221006
Harits Afiq Nugroho	18221012
Justin Yusuf Abidjoko	18221016
Ahmad Rivai Yahya	18221017
Anjani Ibrahim	18221031

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

Daftar Isi

1 Ringkasan

Pada tugas besar kali ini, kami diharuskan untuk melakukan pengembangan robot *video game console* milik Indra dan Doni dengan cara menambahkan fitur serta permainan. Permainan yang ditambahkan berbasis CLI (*command-line interface)* yang berbasis bahasa C. Indra dan Doni memiliki sebuah robot *video game console* yang bernama **BNMO** (dibaca Binomo). Robot *video game console* tersebut sempat mengalami kerusakan sekitar dua bulan yang lalu, namun sudah berhasil diperbaiki. Alat tersebut dapat kembali beroperasi, tetapi terdapat banyak bug yang berada di sistemnya. Pada tugas sebelumnya kami telah berhasil memprogram ulang BNMO. Sekarang, kami diharuskan untuk melakukan pengembangan pada BNMO. Di dalam pengembangan BNMO saat ini, terdapat beberapa fitur utama, diantaranya yaitu memainkan *game*, menambahkan *game*, menghapus *game*, mengurutkan *game* yang akan dimainkan, menampilkan *game* yang telah dimainkan, dan menampilkan *scoreboard game*.

Program utama dari BNMO yang dikembangkan melakukan suatu perintah yang dimana perintah tersebut didapatkan dari pengguna. Setelah pengguna memulai program utama, terdapat pilihan untuk melanjutkan permainan sebelumnya atau mulai dari awal. Lalu terdapat command lainnya yaitu: *save, create game, list game, delete game, queue game, play game, skipgame, quit, help*, *command* lain, *scoreboard*, *reset scoreboard*, *history*, *reset history*, dan command game *rng, diner dash, hangman, tower of hanoi,* dan *snake on meteor* yang dimana jika pengguna memasukkan salah satu dari command tersebut, maka BNMO akan berjalan sesuai *command* yang dimasukkan.

Pada kasus BNMO kali ini, terdapat beberapa ADT yang digunakan yaitu:

1. ADT Stack
2. ADT Set & Map
3. ADT Linked List

Penjelasan mengenai ADT yang digunakan di atas terdapat pada BAB 3 Struktur Data (ADT).

Laporan ini berisikan mengenai penjelasan dari program-program yang kami buat untuk mengembangkan *robot video game console* milik Indra dan Doni. Bagian pertama dari laporan ini berisi tentang ringkasan dari laporan ini beserta gambaran umum mengenai program yang dibuat. Bagian dua menjelaskan mengenai tambahan spesifikasi yang dimana tambahan spesifikasi ini berisi penjelasan fitur yang belum rinci dari Deskripsi Tugas Besar. Bagian tiga menjelaskan tentang struktur data (ADT) yang digunakan pada program ini. Bagian empat menjelaskan program utama dimana bagian ini menjelaskan mengenai algoritma program utama. Bagian lima berisikan algoritma-algoritma menarik yang kami temukan pada pengerjaan tugas besar dua nantinya. Bagian enam menjelaskan tentang pengujian-pengujian data. Bagian tujuh berisikan test script. Bagian delapan menjelaskan tentang deskripsi kerja dalam kelompok. Bagian terakhir berisikan lampiran.

Akhirnya robot video game console BNMO milik Indra dan Doni dapat berkembang dengan lancar sesuai dengan spesifikasi. Seiring dengan berkembangnya BNMO dengan baik, kelompok kami semakin mengerti mengenai penggunaan-penggunaan ADT yang aplikatif. Selain itu kelompok kami juga bekerja sama dengan baik.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 Spesifikasi Fitur Tambahan 1

Fitur tambahan pertama yang ada pada tugas ini adalah tambahan piringan pada game Tower of Hanoi. Pada tugas besar kali ini, piringan yang dapat dimainkan berjumlah berapapun tetapi terdapat jumlah maksimum yaitu sepuluh.

2.2 Spesifikasi Fitur Tambahan 2

Fitur tambahan kedua yang ada pada tugas ini adalah dapat tembusnya *snake* dan terdapat *obstacle* pada game Snake on Meteor. *Snake* dapat tembus keluar dari bidang permainan dan akan muncul pada bidang yang berseberangan. *Obstacle* yang ada akan keluar secara acak di awal game dan tidak akan berpindah seterusnya.

2.3 Spesifikasi Fitur Tambahan 3

Fitur tambahan ketiga yang ada pada tugas ini adalah adanya dua menu ketika memainkan permainan Hangman, yaitu bermain langsung atau menambahkan kata-kata ke dalam kamus yang berada dalam daftar kata dan program dapat membaca list kata dari file lalu setelah permainan selesai, maka *list* kata akan disimpan kembali ke filenya.

3 Struktur Data (ADT)

3.1 ADT Stack

- Sketsa struktur data : Struktur *stack* terdiri atas Stack yang berisikan idxTypeStack yang bertipe *integer* dan infoTypeStack buffer [CAPACITY] yang bertipe *char*.
- Persoalan yang diselesaikan : Memainkan game Tower of Hanoi dan history.
- Alasan pemilihan : Tumpukan piringan pada game Tower of Hanoi direpresentasikan sebagai *stack* yang berurut *last in first out* dalam artian tumpukan terluar yang bisa dipindahkan.
- Diimplementasikan sebagai ADT *stack* dengan nama *file header* “stack.h”.

3.2 ADT Map

- Sketsa struktur data : Struktur *map* terdiri atas Map yang berisikan Elements yang bertipe *pointer to* infotype dan address_M yang bertipe *integer.* Infotype terdiri atas Key yang bertipe keytype serta Value yang bertipe valuetype. Keytype dan valuetype bertipe integer.
- Persoalan yang diselesaikan : Permasalahan pada *scoreboard.*
- Alasan pemilihan : Agar tidak terdapat nilai Key yang sama (nama pemain) di Elements pada Map
- Diimplementasikan sebagai ADT *map* dengan nama *file header* "map.h".

3.3 ADT Linked List

- Sketsa struktur data : Struktur *linked list* terdiri atas Linked List yang dicatat First (elemen pertama list) serta Last (elemen terakhir list). First dan Last bertipe address. Address bertipe *pointer to* ElmtList. ElmtList berisi info yang bertipe point serta next yang bertipe address.
- Persoalan yang diselesaikan : Memainkan game Snake on Meteor.
- Alasan pemilihan : Mengetahui tubuh dari *snake*-nya, serta menyambungkan komponen tubuh snake ketika bergerak dimana posisi pergerakan anggota tubuh akan mengikuti posisi anggota tubuh sebelumnya.
- Diimplementasikan sebagai ADT *linked list* dengan nama *file header* "listlinier.h".

3.4 ADT Point

- Sketsa struktur data : Struktur *point* terdiri atas Point yang berisikan x dan y yang bertipe *integer*.
- Persoalan yang diselesaikan : Memainkan game Snake on Meteor.
- Alasan pemilihan : Penentuan lokasi makanan, obstacle, serta meteor yang akan keluar secara random.
- Diimplementasikan sebagai ADT *point* dengan nama *file header* "point.h"

3.5 ADT Array Dinamik Scoreboard

- Sketsa struktur data : Struktur ArrayDin_SB terdiri atas Capacity dan Neff yang bertipe *integer* serta A yang bertipe *pointer to* ElType_SB. ElType_SB bertipe Map.
- Persoalan yang diselesaikan : Permasalahan pada *scoreboard*.
- Alasan pemilihan : Setiap game memiliki map yang berisi berbagai pasangan nama-skor. ArrayDin_SB menampung pasangan nama-skor pada tiap map untuk setiap game. Selain itu, hal ini dibuat agar nilai-nilai yang berada di *scoreboard* dapat berubah secara dinamis.
- Diimplementasikan sebagai ADT Array Dinamik Scoreboard dengan nama *file header* "arraydin_sb.h".

4 Program Utama

Program utama dengan nama file main.c akan meng-*include file* “console.c” yang meng-*include file header* “console.h”. *File* “console.h” meng-*include file header* “console.h” dan semua *file header* dari ADT yang digunakan. Program utama dimulai dengan memanggil fungsi *start* atau *load*. Kemudian pengguna diminta untuk memasukkan *command* yang sesuai dengan spesifikasi. Terdapat beberapa *command* yaitu CREATE GAME untuk menambahkan game yang dapat ditampilkan dengan fungsi *list*, dan *command* DELETE GAME untuk menghapus *game* dari *list* tetapi game yang dapat dihapus hanya *game* dengan nomor urutan di atas lima dan *game* yang tidak ada di queue. Untuk memainkan game, *user* perlu menambahkan antrean game dengan menggunakan *command* QUEUE GAME dan menggunakan *command* PLAY GAME. Dengan memasukkan *command* PLAY GAME*,* game pertama yang ada di *queue* game akan dimainkan. List dalam *queue* game akan hilang dengan menggunakan *command* QUIT yang juga berarti keluar dari sistem. Untuk melewati permainan yang berada pada antrean *queue,* dapat menggunakan *command* SKIP GAME. Selain itu terdapat *command* HELP yang akan menampilkan penjelasan dari *command-command* yang ada pada sistem. Untuk menyimpan state game pemain saat ini ke dalam suatu *file, user* dapat menggunakan *command* SAVE. Jika *user* ingin melihat riwayat game yang pernah dimainkan, *user* dapat memasukkan *command* HISTORY dan jika ingin menghapus *history* tersebut, *user* dapat memasukkan *command* RESET HISTORY. Jika *user* ingin melihat *score* dari game-game yang pernah dimainkan, *user* dapat melihatnya dengan memasukkan *command* SCOREBOARD dan jika ingin menghapus *scoreboard* tersebut, maka *user* dapat memasukkan *command* RESET SCOREBOARD. Jika input dari *user* merupakan *command* yang tidak dikenali, maka *user* akan diminta memasukkan input yang valid lagi. Namun, jika input dari *user* kurang lengkap, maka sistem akan mengeluarkan saran untuk melengkapi *command* yang kurang lengkap tersebut.

Secara *default*, terdapat lima game yang dapat dimainkan yaitu RNG, Diner Dash, Tower of Hanoi, Snake on Meteor, dan Hangman. RNG merupakan game menebak angka sampai angka yang ditebak *user* sama dengan angka yang ditentukan program, Diner Dash merupakan game dengan konsep restoran dengan pengantaran makanan berdasarkan prioritasnya, Tower of Hanoi merupakan game yang mengharuskan *user* untuk memindahkan tumpukan piringan sesuai dengan peraturan yang ada, Snake on Meteor merupakan game yang dimana *user* akan mengumpulkan makanan sebanyak mungkin agar ular dapat bertambah panjang dan tidak melanggar peraturan yang ada, dan Hangman adalah permainan yang dimana *user* diharuskan untuk menebak kata sesuai dengan yang telah ditentukan oleh program.

5 Algoritma-Algoritma Menarik

5.1 Algoritma 1

Algoritma menarik pertama yang dibuat pada tugas besar kali ini adalah ADT Array Dinamik Scoreboard. Algoritma ini digunakan pada pemrograman *scoreboard* yang berfungsi agar jumlah *scoreboard* dapat berubah secara dinamis. Algoritma ini dianggap menarik karena jika tidak ada ADT ini, jumlah *scoreboard* tidak dapat berubah secara dinamis ketika membuat *game* baru.

5.2 Algoritma 2

Algoritma menarik kedua yang dibuat pada tugas besar kali ini adalah membentuk piringan dengan cara memanipulasi string. Algoritma ini diterapkan pada permainan Tower of Hanoi guna untuk menampilkan piringan. Algoritma ini dianggap menarik karena dapat menampilkan piringan secara urut mengecil ke atas dan ketika memindahkan piringan, bentuk dari piringan tersebut tetap presisi.

5.3 Algoritma 3

Algoritma menarik ketiga yang dibuat pada tugas besar kali ini adalah algoritma yang memungkinkan program dapat berjalan di dua sistem operasi yang berbeda (Linux dan Windows. Algoritma ini dianggap menarik karena adanya perbedaan cara kerja program ketika dijalankan pada sistem Windows dan Linux. Untuk menyelesaikan masalah ini, diperlukan #ifdef #else #endif di *file* console.

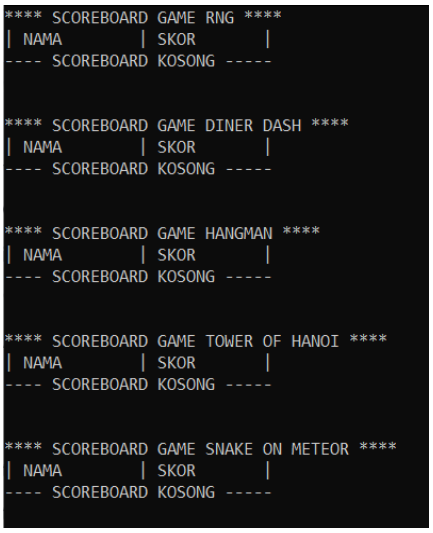
5.4 Algoritma 4

Algoritma menarik keempat yang dibuat pada tugas besar kali ini adalah melakukan *print* terhadap arena permainan pada game Snake On Meteor berdasarkan kumpulan *string* yang disimpan pada suatu variabel bertipe data *pointer to* char*. Setiap titik <x,y> dapat ditranslasikan pada *index* ke x + y * 5 pada variabel yang menyimpan kumpulan *string* tersebut. Nilai pada *index* tersebut disesuaikan dengan keberadaan makanan, meteor, obstacle, kepala *snake*, anggota tubuh selain kepala pada *snake*, serta ketidakberadaan apapun pada suatu titik.

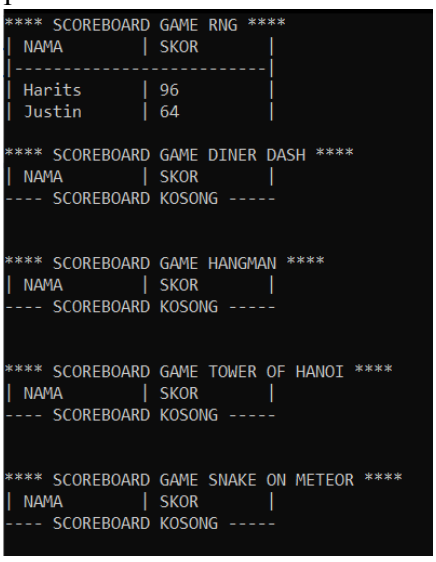
6 Data Test

6.1 Data Test 1

Tes ini dilakukan untuk memastikan bahwa program sudah dapat menampilkan *scoreboard* sesuai dengan spesifikasi.



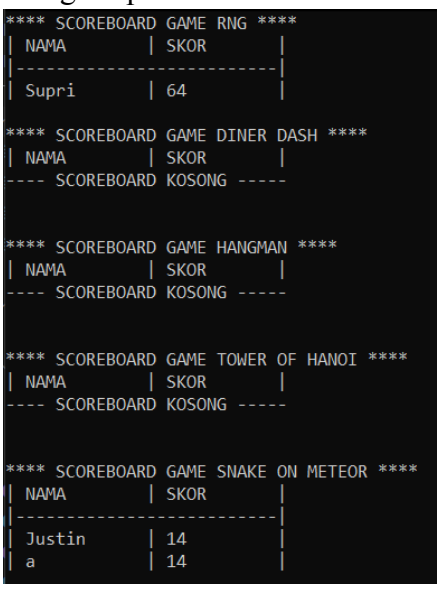
Gambar 6.1.1 Tampilan Awal Ketika Belum Memainkan Game Apapun



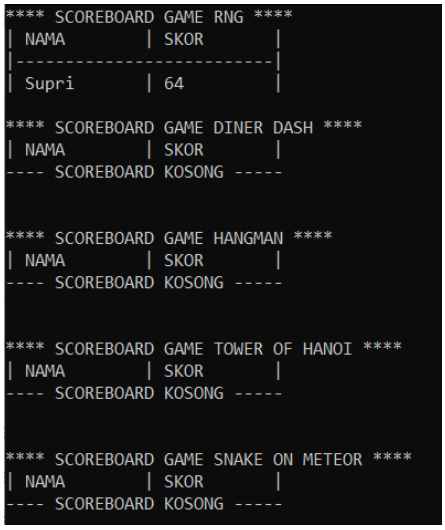
Gambar 6.1.2 Tampilan Ketika Sudah Ada Beberapa Score Permainan

6.2 Data Test 2

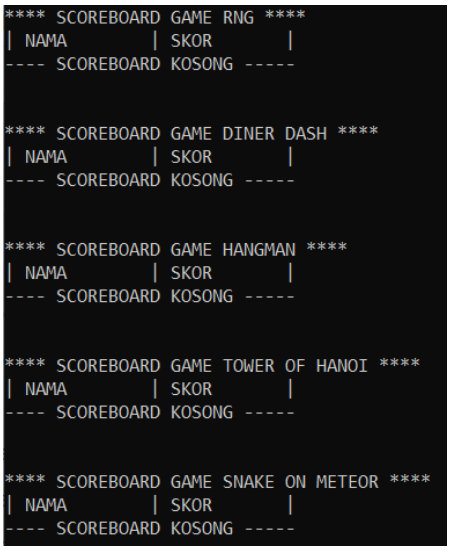
Tes ini dilakukan untuk memastikan bahwa program sudah dapat me-*reset scoreboard* yang telah ada sesuai dengan spesifikasi.



Gambar 6.2.1 Tampilan Scoreboard Awal



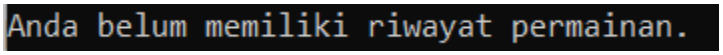
Gambar 6.2.2 Tampilan Scoreboard Ketika Melakukan Reset Scoreboard Game Snake on Meteor



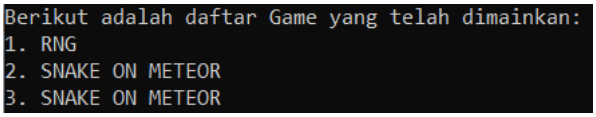
Gambar 6.2.3 Tampilan Scoreboard Ketika Melakukan Reset Scoreboard All

6.3 Data Test 3

Tes ini dilakukan untuk memastikan bahwa program sudah dapat menampilkan *history* sesuai dengan spesifikasi.



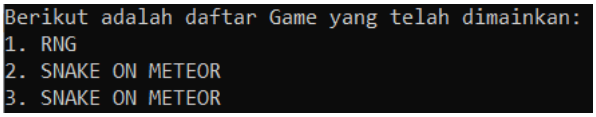
Gambar 6.3.1 Tampilan History Jika Belum Memainkan Game



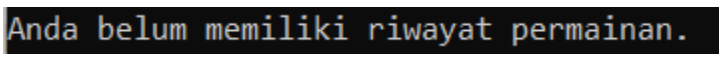
Gambar 6.3.2 Tampilan History Setelah Memainkan Game yang Ada Pada Poin 6.2

6.4 Data Test 4

Tes ini dilakukan untuk memastikan bahwa program sudah dapat me-*reset history* yang sudah ada sesuai dengan spesifikasi.



Gambar 6.4.1 Tampilan History Awal



Gambar 6.4.2 Tampilan History Setelah Melakukan Reset

6.5 Data Test 5

Tes ini dilakukan untuk memastikan bahwa program sudah dapat memainkan Hangman sesuai dengan spesifikasi.

```
Loading HANGMAN ...

================================== HANGMAN ==================================
Hangman merupakan permainan menebak kata dengan menebak satu-satu huruf yang ada
pada kata. Pemain memiliki 10 kesempatan dalam bermain. Jika pemain berhasil
menebak kata, maka pemain akan mendapatkan poin sesuai panjang kata yang
berhasil ditebak, dan permainan akan berlanjut sampai kesempatan habis.

Pilihan menu:
  1. Bermain langsung
  2. Menambahkan kata-kata ke dalam sistem
Masukkan pilihan menu:
```

Gambar 6.5.1 Tampilan Awal Ketika Memasuki Permainan Hangman

```
Selamat bermain!

Kategori: Barang
Tebakan sebelumnya: -

_ _ _ _
Kesempatan: 10
Masukkan tebakan:
```

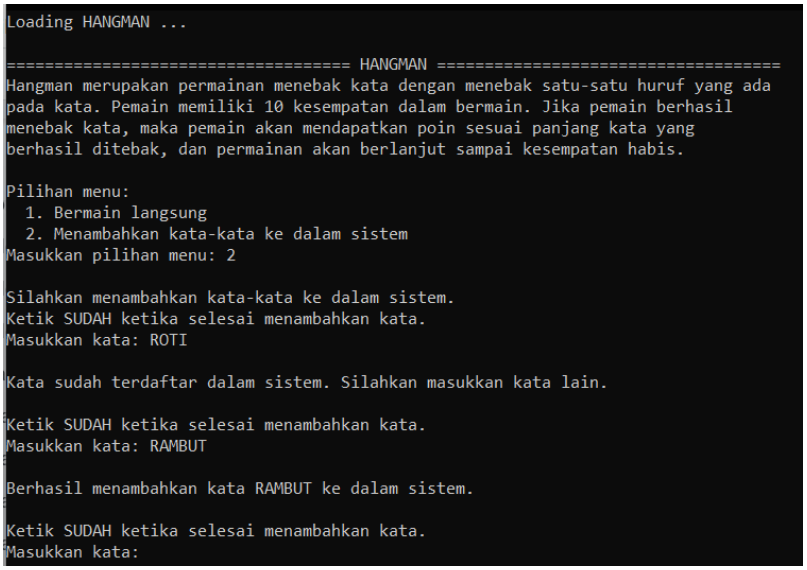
Gambar 6.5.2 Tampilan Awal Ketika Memilih Opsi Satu pada Menu Sebelumnya

```
Tebakan sebelumnya: A I U K L T R M N B H
B A _ U
Kesempatan = 2
Masukkan tebakan: R

Tebakan sebelumnya: A I U K L T R M N B H R
B A _ U
Kesempatan = 1
Masukkan tebakan: E

Tebakan sebelumnya: A I U K L T R M N B H R E
B A _ U
Kesempatan = 0
Skor akhir: 0
Nama (Max. 11 karakter): _
```

Gambar 6.5.3 Tampilan Ketika Berhasil Menebak Kata dan Game Berakhir

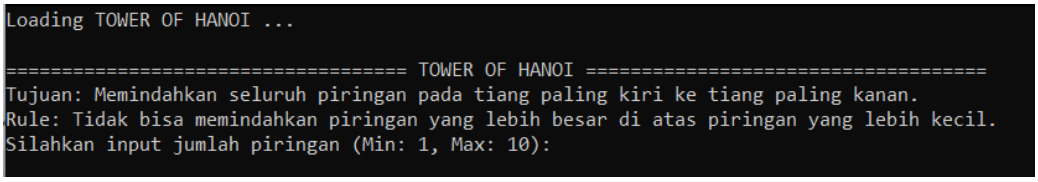


Gambar 6.5.4 Tampilan Ketika Memilih Opsi Dua pada Menu Sebelumnya

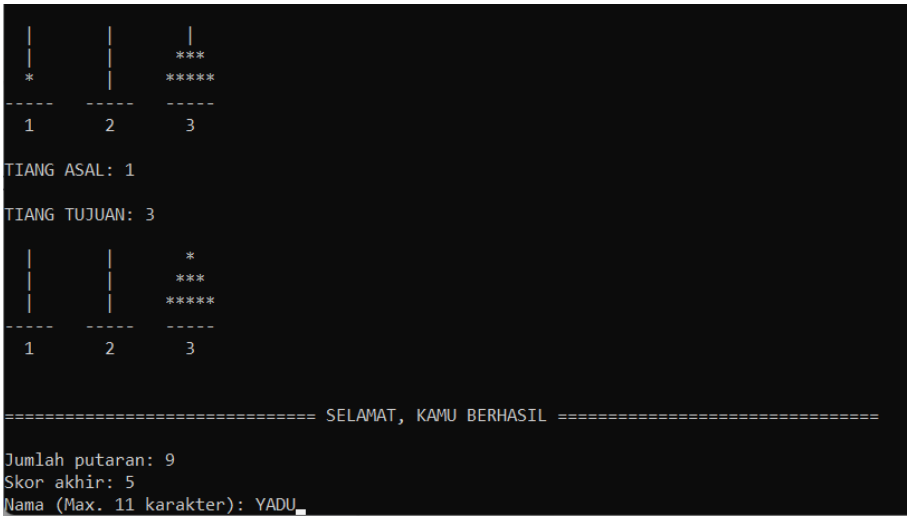
Ketika *user* memasukkan kata "SUDAH", maka otomatis permainan Hangman akan langsung dimulai.

6.6 Data Test 6

Tes ini dilakukan untuk memastikan bahwa program sudah dapat memainkan Tower of Hanoi sesuai dengan spesifikasi.



Gambar 6.6.1 Tampilan Awal Ketika Memasuki Permainan Tower of Hanoi



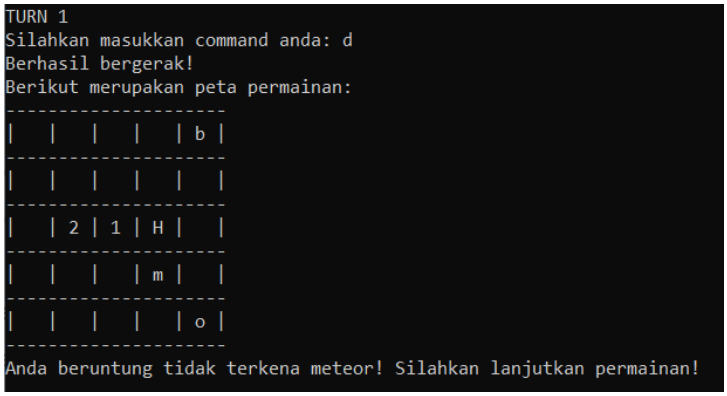
Gambar 6.6.2 Tampilan Ketika Memainkan dan Game Berakhir

6.7 Data Test 7

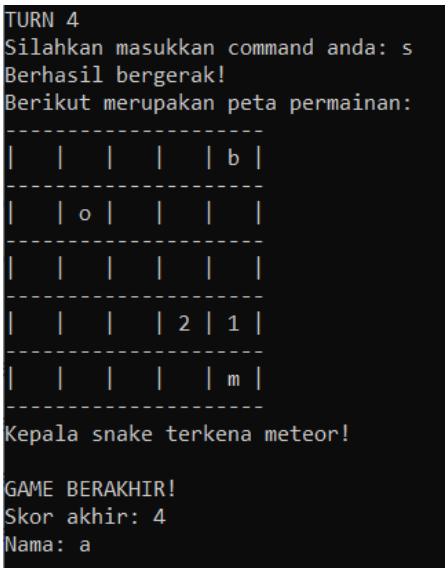
Tes ini dilakukan untuk memastikan bahwa program sudah dapat memainkan game Snake on Meteor sesuai dengan spesifikasi.

Gambar 6.7.1 Tampilan Awal Ketika Memainkan Snake on Meteor

.



Gambar 6.7.2 Tampilan Setelah Melakukan Gerakan Pertama



Gambar 6.7.3 Tampilan Akhir Permainan

7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Fitur SCOREBOARD	Memeriksa apakah fitur dapat menampilkan *scoreboard*	Memasukkan *command* SCOREBOARD pada menu utama	Data *Test* 1	Dapat menampilkan *scoreboard* sesuai dengan spesifikasi	Sesuai yang diharapkan
2	Fitur RESET SCOREBOARD	Memeriksa apakah fitur dapat menghapus *scoreboard* sesuai dengan input yang diinginkan	Memasukkan *command* RESET SCOREBOARD pada menu utama lalu memasukkan *command* game apa yang akan di-*reset scoreboard*-nya dan yang terakhir konfirmasi ulang jika akan melakukan *reset*	Data *Test* 2	Penghapusan data pada *scoreboard* dapat dilakukan sesuai dengan spesifikasi	Sesuai yang diharapkan
3	Fitur HISTORY	Memeriksa apakah fitur dapat menampilkan *history*	Memasukkan *command* HISTORY pada menu utama	Data *Test* 3	Dapat menampilkan *history* sesuai dengan spesifikasi	Sesuai yang diharapkan
4	Fitur RESET HISTORY	Memeriksa apakah fitur dapat menghapus *history* sesuai dengan input yang diinginkan	Memasukkan *command* RESET HISTORY pada menu utama lalu melakukan konfirmasi ulang jika akan melakukan *reset*	Data *Test* 4	Penghapusan data pada *history* dapat dilakukan sesuai dengan spesifikasi	Sesuai yang diharapkan
5	Fitur HANGMAN	Memeriksa apakah fitur dapat berjalan dengan memainkan Hangman sesuai dengan spesifikasi	Memasukkan *command* QUEUE GAME (Memilih Hangman) → PLAY GAME pada menu utama lalu menjalankan permainan sesuai dengan spesifikasi	Data *Test* 5	Dapat memainkan game Hangman sesuai dengan spesifikasi	Sesuai yang diharapkan
6	Fitur TOWER OF HANOI	Memeriksa apakah fitur dapat berjalan dengan memainkan Tower of Hanoi sesuai	Memasukkan *command* QUEUE GAME (Memilih Tower of Hanoi) → PLAY GAME pada menu utama lalu menjalankan permainan sesuai dengan spesifikasi	Data *Test* 6	Dapat memainkan game Tower of Hanoi sesuai dengan spesifikasi	Sesuai yang diharapkan

		dengan spesifikasi				
7	Fitur SNAKE ON METEOR	Memeriksa apakah fitur dapat berjalan dengan memainkan Snake on Meteor sesuai dengan spesifikasi	Memasukkan *command* QUEUE GAME (Memilih game Snake on Meteor) → PLAY GAME pada menu utama lalu menjalankan program sesuai dengan spesifikasi	Data *Test* 7	Dapat memainkan game Snake on Meteor sesuai dengan spesifikasi	Sesuai yang diharapkan

8 Pembagian Kerja dalam Kelompok

Nama Anggota Kelompok	**Pembagian Kerja**
Richard Haris	● Membuat fungsi: 1. SCOREBOARD 2. SNAKEONMETEOR serta 2 bonusnya 3. Beberapa fungsi pendukung, seperti mod, int_to_string, addSpace, MakeSnake, PrintArena, SnakeBody. ● Membuat dan memodifikasi ADT (file header(.h), implementasi(.c), serta driver): 1. ArrayDin_SB, Map, List Linier, serta Point ● Membantu membuat laporan bagian Struktur Data (ADT) serta algoritma-algoritma menarik
Harits Afiq Nugroho	● Membuat fungsi: 1. HISTORY 2. RESET HISTORY 3. Mengembangkan HELP ● Membuat laporan
Justin Yusuf Abidjoko	● Membuat fungsi : 1. RESET SCOREBOARD 2. Mengembangkan LOAD 3. Mengembangkan SAVE

	4. Bonus TOWEROFHANOI & HANGMAN 5. Beberapa fungsi tambahan untuk program utama yang tidak dicantumkan dalam spesifikasi tugas (upper, lower, splitStringInt, inputNamaScoreboard, power) ● Membuat program berjalan di Linux ● Memberi arahan pembagian tugas serta cara pengerjaan tugas ● Menggabungkan code fungsi dari anggota kelompok
Ahmad Rivai Yahya	● Membuat fungsi: 1. TOWEROFHANOI ● Membantu menyempurnakan laporan
Anjani Ibrahim	● Membuat fungsi: 1. HANGMAN ● Membantu membuat laporan

9 Lampiran

9.1 *Deskripsi Tugas Besar 2*

1. Command

Terdapat beberapa aturan umum command yang digunakan:

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

a. Command Dasar

Semua command yang terdapat pada Spesifikasi Tugas Besar 1 IF2111 2022/2023.

b. SCOREBOARD

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

```
// Misal tahap ini adalah tahap game over dari game
// EIFFEL TOWER
Skor akhir: 12
Nama: BNMO
```

```
ENTER COMMAND: SCOREBOARD
**** SCOREBOARD GAME TOWER OF HANOI ****
| NAMA        | SKOR       |
|-------------------------|
| BNMO        | 12         |
| Finn        | 9          |

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

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

```
| NAMA        | SKOR       |
|-------------------------|
| BNMO        | 80         |
| Finn        | 62         |

**** SCOREBOARD GAME RNG****
| NAMA        | SKOR       |
|-------------------------|
| BNMO        | 13         |
| Finn        | 11         |

**** SCOREBOARD GAME HANGMAN****
| NAMA        | SKOR       |
|-------------------------|
| BNMO        | 99         |
| Finn        | 1          |
```

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*.

```
ENTER COMMAND: RESET SCOREBOARD

DAFTAR SCOREBOARD:
0. ALL
1. RNG
2. Diner DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR

SCOREBOARD YANG INGIN DIHAPUS: 0

APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD ALL
(YA/TIDAK)? YA

Scoreboard berhasil di-reset.
```

a

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.

```
ENTER COMMAND: HISTORY 2
Berikut adalah daftar Game yang telah dimainkan
1. EIFFEL TOWER
2. RNG
```

```
ENTER COMMAND: HISTORY 8
Berikut adalah daftar Game yang telah dimainkan
1. EIFFEL TOWER
2. RNG
3. EIFFEL TOWER
4. RISEWOMAN
5. LUNCH SLOW
```

e. RESET HISTORY

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

```
ENTER COMMAND: RESET HISTORY

APAKAH KAMU YAKIN INGIN MELAKUKAN RESET HISTORY? YA

History berhasil di-reset.
```

```
ENTER COMMAND: RESET HISTORY

APAKAH KAMU YAKIN INGIN MELAKUKAN RESET HISTORY? TIDAK

History tidak jadi di-reset. Berikut adalah daftar Game yang
telah dimainkan
1. EIFFEL TOWER
2. RNG
3. EIFFEL TOWER
4. RISEWOMAN
5. LUNCH SLOW
```

a. RNG

Dijelaskan pada Spesifikasi Tugas Besar 1 IF2111 2022/2023.

b. Diner Dash

Dijelaskan pada di Spesifikasi Tugas Besar 1 IF2111 2022/2023.

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

- Contoh output sebuah permainan Hangman (**tampilan tidak harus ditiru, tampilannya sangat dibebaskan dan boleh sekreatif mungkin)** :

Tebakan sebelumnya: -
Kata: _ _ _ _
Kesempatan: 10
Masukkan tebakan: **A**

Tebakan sebelumnya: a
Kata: _ A _ A
Kesempatan: 10
Masukkan tebakan: **b**

Tebakan sebelumnya: ab
Kata: _ A _ A
Kesempatan: 9
Masukkan tebakan: **M**

Tebakan sebelumnya: abm
Kata: M A _ A
Kesempatan: 9
Masukkan tebakan: **F**

Tebakan sebelumnya: abmf
Kata: M A _ A
Kesempatan: 8
Masukkan tebakan: **t**

Berhasil menebak kata MATA! Kamu mendapatkan 4 poin!

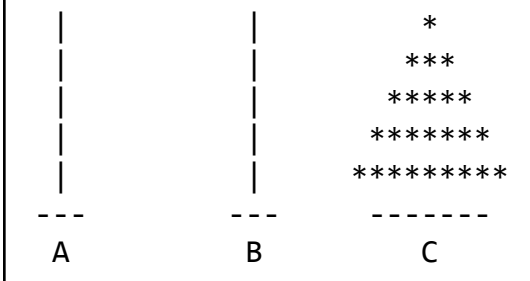
Tebakan sebelumnya: -
Kata: _ _ _ _ _ _
Kesempatan: 8
Masukkan tebakan:

Permainan berlanjut hingga kesempatan habis

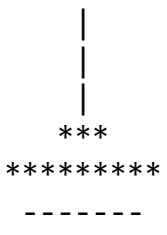
d. Tower of Hanoi

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, BNMO melihat panduannya di The Enchiridion. Setelah itu, BNMO ingin melakukan modifikasi permainan ini:

- Jumlah piringan hanya 5 saja untuk permainan ini.
- Piringan direpresentasikan sebagai gambar piringan dengan *, dengan piringan paling besar adalah 9 dan balok silinder paling kecil adalah 1.
- Skor untuk permainan ini tergantung dengan seberapa optimal langkah dari pemain (dengan langkah paling optimalnya adalah 31) dan skor maksimalnya adalah 10 (Cara perhitungan skornya dibebaskan, yang terpenting adalah jika langkahnya 31, skornya adalah 10).
- Contoh (**tampilan tidak harus ditiru, tampilannya sangat dibebaskan dan boleh sekreatif mungkin**):

Awal	Akhir
	
TIANG ASAL: **A** TIANG TUJUAN: **B** Memindahkan piringan ke B...	Kamu berhasil! Skor didapatkan: 10 Nama: **BNMO**

- Contoh jika tiang hanya terisi sebagian.



e. Snake on Meteor

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

Untuk pemahaman spek tubes, kosakata ini akan digunakan:

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

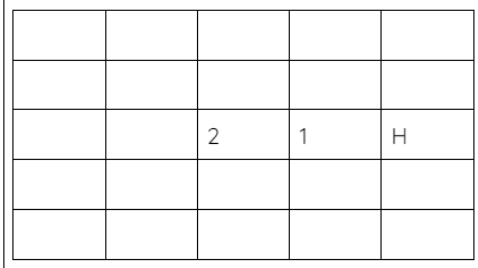
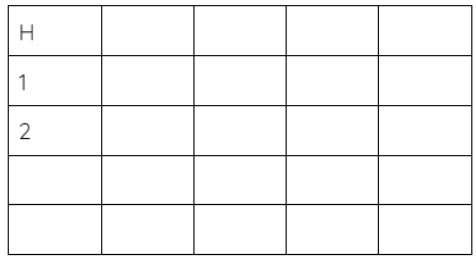
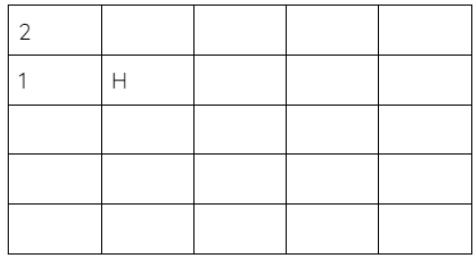
Berikut ini merupakan spesifikasi yang lebih *detail* terkait game *snake on meteor:*

- **Dimensi Peta**: Dimensi peta adalah 5x5 unit dengan <0,0> merupakan sisi kiri atas dan <4,4> sisi kanan bawah. Sistem koordinat untuk penjelasan spek ini adalah:

<0,0>				<4,0>
<0,4>				<4,4>

- **Panjang snake**: Panjang awal dari *snake* adalah 3 unit. Kepala dari *snake* di-*random* pada sebuah titik dan 2 anggota badan yang berurut menurun dengan prioritas horizontal yang sama. Apabila badan menabrak dinding, maka akan berurut menurun dengan prioritas vertical yang sama. **Maksud**

dari menurun adalah secara skalar (Cth: Dari angka 3 ke angka 2, dari angka 2 ke 1, dst)

Penjelasan	Visualisasi
Kepala H: <4,2> Badan 1: <3,2> Badan 2: <2,2>	
Kepala H: <0,0>	
Badan 1: <0,1> Badan 2: <0,2>	
Kepala H: <1,1> Badan 1: <0,1> Badan 2: <0,0>	

- **Makanan**: Makanan akan diberikan secara random pada sebuah titik <x,y> (ditandai dengan huruf **o**). Lokasi munculnya makanan tidak mungkin berada di titik yang sama dengan komponen tubuh *snake*. Jika berhasil dimakan oleh *snake,* maka *snake* akan langsung bertambah panjang ekornya sebanyak 1 unit dan makanan baru akan di-*random* lagi pada sebuah titik <x,y>

  Proses pertambahan tail adalah sebagai berikut:

  a. **Secara umum**, ekor *snake* akan bertambah pada titik ordinat yang sama dengan tail, tetapi dengan titik koordinat satu sebelum tailnya

(**Apabila tail berada pada titik <x,y>, maka pertambahan tail akan dilakukan pada titik <x-1,y>**).

b. **Apabila kasus a tidak mungkin** (misalkan karena tail berada pada titik <0,1> dan tidak memungkinkan ada nilai titik <-1,1>), maka pertambahan akan dilakukan pada titik ordinat satu sebelum ekor *snake* sekarang, tetapi pada titik koordinat yang sama (**Apabila tail berada pada titik <x,y>, maka pertambahan tail akan dilakukan pada titik <x,y-1>**)

c. **Apabila kasus b tidak mungkin** (misalkan karena ekor berada pada titik <0,0> dan tidak memungkinkan adanya nilai <-1,0> ataupun <0,-1>), maka pertambahan akan dilakukan pada titik ordinat satu setelah ekor *snake* sekarang, tetapi pada titik koordinat yang sama (**Apabila tail berada pada titik <x,y>, maka pertambahan tail akan dilakukan pada titik <x,y+1>**)

d. **Apabila kasus a,b dan c tidak mungkin** (misalkan karena ekor berada pada titik <0,0> dan tidak memungkinkan adanya nilai <-1,0> ataupun <0,-1> serta terdapat anggota tubuh pada titik <0,1>), maka pertambahan akan dilakukan pada titik ordinat yang sama dengan ekor *snake* sekarang, tetapi pada titik koordinat yang bertambah satu (**Apabila tail berada pada titik <x,y>, maka pertambahan tail akan dilakukan pada titik <x+1,y>**)

e. **Apabila kasus a,b,c,d tidak mungkin** (misalkan ekor berada pada titik <2,2> dan terdapat anggota tubuh di titik <1,2>,<2,1>,<2,3> dan <3,2>) maka game akan berakhir

- **Cara bergerak**: *Game* setiap putarannya akan meminta pemain untuk memasukkan huruf ‘a’, ’w’, ’s’ atau ‘d’ untuk menggerakan kepala *snake* (*game* akan meminta *re-input* apabila masukan selain huruf tersebut. Spesifikasi *lowercase* atau *uppercase* dibebaskan kepada kalian). Huruf ‘a’ untuk menggerakan kepala ke kiri, ’w’ untuk menggerakan kepala ke atas, ‘s’ untuk menggerakan kepala ke bawah dan ‘d’ untuk menggerakan kepala ke kanan. Setiap pergerakan akan menambahkan nilai koordinat/ordinat kepala *snake* (tergantung *input* yang diberikan) sebanyak 1 unit. **Posisi pergerakan anggota tubuh akan mengikuti posisi anggota tubuh sebelumnya. Kepala *snake* tidak mungkin bergerak ke anggota tubuhnya sendiri(game akan meminta input ulang apabila hal tersebut terjadi)**

Contoh:

a. Turn 0: Kepala *snake* berada pada titik <4,2>(titik H) dengan badannya berada pada <3,2>(titik 1) dan <2,2>(titik 2) secara berurutan (maka *snake* memiliki urutan <4,2>,<3,2>,<2,2>)

<0,0>				<4,0>
		2	1	H
<0,4>				<4,4>

b. Turn 1: Pemain memberikan masukan berupa 'w'. Posisi kepala *snake* akan berada pada titik <4,1>. Sekarang bagian badan akan berpindah, maka anggota badan pada <3,2> akan berpindah ke <4,2>(mengikuti posisi head pada turn sebelumnya) dan anggota badan pada <2,2> akan berpindah ke <3,2>(mengikuti posisi titik 1 pada turn sebelumnya)

<0,0>				<4,0>
				H
			2	1
<0,4>				<4,4>

c. Turn 2: Pemain memberikan masukan berupa 'a'. Posisi kepala *snake* akan berada pada titik <3,1> dan anggota badan akan berada pada <4,1> dan <4,2>

<0,0>				<4,0>
			H	1
				2
<0,4>				<4,4>

- **Meteor**: Setiap putaran setelah permainan berhasil digenerate(turn >1), 1 meteor akan di-*random* pada titik tertentu(ditandai dengan huruf **m)**. Apabila salah satu bagian dari *snake* terkena meteor, maka bagian tersebut akan dihapus dari *snake* dan panjang dari *snake* akan berkurang sebanyak 1. Apabila komponen dari *snake* (kepala/badan/ekor) terkena meteor (akan disebut *hit* untuk mempermudah pemahaman kalian), maka bagian badan sebelum *hit* akan tersambung dengan bagian badan setelah *hit.* Setelah terkena *hit*, ada kemungkinan badan *snake* berada di koordinat yang saling diagonal. Selanjutnya, **makanan tidak dapat muncul di titik ini dan kepala snake juga tidak bisa mengunjungi titik ini di turn selanjutnya**.

  Contoh: *Snake* memiliki anggota tubuh <3,3> (titik H), <2,3> (titik 1), <1,3> (titik 2), <0,3> (titik 3) dan meteor menyerang peta pada titik 2. Maka bagian badan <1,3> akan dihapus dan sekarang anggota tubuh berupa <3,3>,<2,3>,<0,3>

Sebelum terkena meteor	<0,0> · · · <4,0> · · · · · · · · · · 3 2 1 H · <0,4> · · · <4,4>
Saat terkena meteor	<0,0> · · · <4,0> · · · · · · · · · · 3 m 1 H · <0,4> · · · <4,4>

Setelah terkena meteor	<0,0> · · · <4,0> · · · · · · · · · · 2 · 1 H · <0,4> · · · <4,4>

- **Kondisi Menang**: Tidak terdapat kondisi menang secara khusus. *Game* berakhir ketika terjadi kekalahan. Pada akhir game, satu unit pada komponen *snake* akan dikonversi menjadi 2 *point.*

  Contoh: Pemain kalah dengan panjang akhir *snake* 10 unit, maka *score* yang didapat ialah 20 *point*
- **Kondisi Kalah**: Terdapat beberapa kondisi kekalahan dari *game* ini, yaitu
  1. Seluruh komponen *snake* (kepala, badan, ekor) terkena meteor → panjang *snake* adalah **0**
  2. Kepala dari *snake* terkena meteor → panjang snake adalah **panjang badan**

3. Ekor baru tidak dapat di-*spawn* karena tidak dapat area di kiri, atas, bawah ataupun kanan ekor lama → panjang *snake* adalah **panjang badan sebelum ekor baru ditambahkan**

- **Contoh Permainan:**

Selamat datang di snake on meteor!

Mengenerate peta, snake dan makanan . . .

Berhasil digenerate!

Berikut merupakan peta permainan

			o	
2	1	H		

TURN 1:
Silahkan masukkan command anda: **haha**
Command tidak valid! Silahkan input command menggunakan huruf w/a/s/d

**catatan: karena input tidak valid, dilakukan validasi dan turn tidak bertambah*

/*contoh kasus pergerakan normal*/
TURN 1:
Silahkan masukkan command anda: **w**

Berhasil bergerak!
Berikut merupakan peta permainan:

		H	o	
	2	1		
			m	

Anda beruntung tidak terkena meteor! Silahkan lanjutkan permainan

/*contoh kasus berhasil makan*/
TURN 2:
Silahkan masukkan command anda: d

Berhasil bergerak!
Berikut merupakan peta permainan

				o
	m	1	H	
	3	2		

Anda beruntung tidak terkena meteor! Silahkan lanjutkan permainan

/*contoh kasus terkena meteor pada titik 1*/
TURN 3:
Silahkan masukkan command anda: **w**

Berhasil bergerak!
Berikut merupakan peta permainan

			H	o
		2	m	
		3		

**catatan: titik 1 terkena meteor sehingga titik satu langsung dihapuskan*

Anda terkena meteor!
Berikut merupakan peta permainan sekarang:

			H	o
		1	m	
		2		

Silahkan lanjutkan permainan

/*contoh kasus ingin pergi ke titik yang terkena meteor pada turn sebelumnya*/
TURN 4:
Silahkan masukkan command anda: **s**

Meteor masih panas! Anda belum dapat kembali ke titik tersebut.
Silahkan masukkan command lainnya

**catatan: karena pada turn 3 titik <3,1> terkena meteor, maka pada turn 4 titik tersebut belum dapat dikunjungi(namun pada turn 5 dan seterusnya sudah dapat dikunjungi kembali)*

TURN 4
Silahkan masukkan command anda: **d**

Berhasil bergerak!
Berikut merupakan peta permainan

			1	H
	3	2	o	
	m			

**catatan: ingat pergerakan anggota tubuh akan selalu mengikuti posisi anggota tubuh sebelumnya(titik 1 akan bergerak ke posisi titik H di turn sebelumnya, titik 2 akan ke titik 1, titik 3 akan ke titik 2,dst)*

/*contoh kasus bergerkan ke diri sendiri*/
TURN 5
Silahkan masukkan command anda: **a**

Anda tidak dapat bergerak ke tubuh anda sendiri!
Silahkan input command yang lain

TURN 5
Silahkan masukkan command anda: **s**

Berhasil bergerak!

Berikut merupakan peta permainan

m			2	1
		3	o	H

/*contoh kasus berhasil makan sekaligus terkena meteor pada kepala*/
TURN 6
Silahkan masukkan command anda: **a**

Berhasil bergerak!
Berikut merupakan peta permainan

		4	3	2
			m	1

Kepala snake terkena meteor!
Game berakhir. Skor: 8

**catatan: Skor dihitung dari panjang sisa anggota tubuh yang dimiliki. Karena pada titik kepala terkena meteor, maka titik tersebut tidak akan dihitung ke dalam sisa panjang snake*

- **Catatan:**
    1. Perhatikan dan validasi *input* gerak dari *snake*
       Contoh: Kepala dari *snake* berada pada titik <4,3> dan badannya <3,3>,<2,3> (posisi *snake* secara berurutan: <4,3>,<3,3>,<2,3>). Maka pemain tidak dapat memberikan *input* ‘a’ pada gerak *snake,* karena akan mengakibatkan kepala *snake* bergerak menuju badan-nya sendiri
    2. Penggambaran peta dibebaskan, boleh menggunakan tabel dengan sekat-sekat atau tabel tanpa sekat-sekat
    3. Gunakan prinsip “apabila ekor tidak dapat *spawn* di-kiri, maka akan dilakukan *spawn* di atas. Apabila di kiri dan di atas tidak mungkin, *spawn* ekor di bawah. Apabila tidak mungkin *spawn* di kiri, atas dan bawah, maka lakukan *spawn* di kanan. Apabila tidak memungkinkan untuk *spawn,* maka akan *game over*”
    4. Peta **tidak harus** ‘tembus’ atau muncul pada sisi sebaliknya apabila dilewati oleh *snake.*

Contoh: Dengan dimensi 5x5, maka titik minimal dari peta adalah <0,0> dan titik maksimal dari peta adalah <4,4>. Apabila kepala berada pada titik <0,0> dan dilakukan *input* ‘a’, maka pada putaran berikutnya kepala tidak harus berada pada titik <4,0>

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

f. 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*

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

No. Kelompok/Kelas : 05/03
Nama Kelompok :
Anggota Kelompok (Nama/NIM) :
1. Richard Haris/18221006
2. Harits Afiq Nugroho/18221012
3. Justin Yusuf Abidjoko/18221016
4. Ahmad RIvai Yahya/18221017
5. Anjani Ibrahim/18221031

Asisten Pembimbing : Aditya Bimawan

Asistensi I

Tanggal : 24 November 2022 **Tempat : Google Meet**	**Catatan Asistensi:** ● Untik di history apakah nambahin nama duplikat? Iya sesuai contoh yang ada ● Untuk scoreboard membuat satu set khusus setiap game, berarti total ada lima set? Iya bisa lebih juga nyesuain jumlah game. ● Untuk create game apakah boleh disimpan di satu set saja jadi hanya ada enam set saja? Kayaknya engga jadi dibuat array yang dinamis karena nama game juga unik. ● Saran: fitur-fitur ada beberapa dan tingkat kesusahan berbeda-beda sehingga untuk pembagian tugas diusahakan saling membantu karena yang tingkat kesusahannya paling besar bobotnya besar. Sayang jika tidak diselesaikan.

Kehadiran Anggota Kelompok: 1 18221006  2 18221012  3 18221016  4 18221017  5 18221031 	● Untuk tower of hanoi bonusnya apakah ada batasan piringan? Tanya di QNA saja agar semua kelompok bisa melihat tapi dari kak Adit dibebasin saja. ● Jika bikin bonus dan tidak tepat apakah mengurangi nilai? tidak mengurangi. ● Konsep meteor di game snake → Meteor akan ngilangin bagian dari snake. Jika melihat contoh, bagian yang terkena meteor akan hilang. Konsepnya adalah link list. index yang terkena meteor akan hilang. Untuk posisi yang badannya hilang, posisi badan setelahnya akan melompat menyambung. Badan ular akan bolong hingga badan yang selanjutnya melewati meteor. Kalau setelah terkena meteor memakan makanan, lalu seperti apa? sesuaikan sesuai contoh. Memang agak aneh tetapi menguatkan konsep mengenai link list. Gridnya tidak harus memakai ADT link list hanya badannya saja.
	Tanda Tangan Asisten: 

Asistensi II

Tanggal : 1 Desember 2022 **Tempat : Google Meet** **Kehadiran Anggota Kelompok:** 1 18221006  2 18221012  3 18221016  4 18221017  5 18221031 	**Catatan Asistensi:** ● Apakah boleh menggunakan header yang bertipe .c? untuk tubes kali ini boleh ● Selain itu (driver, laporan, dll) aman ● Untuk gambar yang ada di spesifikasi tugas apakah perlu dimasukkan kedalam laporan bagian lampiran? Jika halaman masih cukup sebaiknya dimasukkan.
	Tanda Tangan Asisten: 

9.3 *Log Activity Anggota Kelompok*

No.	Waktu	Keterangan
1.	18 November 2022	Pembagian tugas
2.	19 November - 1 Desember 2022	Pengerjaan dan menyatukan program berdasarkan tugas masing-masing
3.	24 November 2022	Asistensi pertama
4.	29 November - 1 Desember	Pembuatan laporan
5.	1 Desember 2022	Asistensi kedua dan menyempurnakan program
6.	2 Desember 2022	Finalisasi akhir dan pengumpulan tugas