

Spesifikasi Tugas Besar 2

IF2111 Algoritma dan Struktur Data STI

BNMO

Revisi

ver. 18 November 2022

ver. 20 November 2022

Isi patch:

- Revisi typo

ver. 23 November 2022

Isi patch:

- Hash Map

ver. 24 November 2022

Isi patch:

- Penjelasan lebih lanjut terkait cara gerak *snake on meteor*
- Penjelasan lebih lanjut terkait cara *spawn* ekor *snake on meteor*
- Perbaikan kondisi kalah *snake on meteor*

ver. 25 November 2022

Isi patch:

- Perubahan file konfigurasi yang di konfigurasi yang disimpan oleh sistem

Deadline

2 Desember 2022, 21:11 WIB

Daftar Isi

Daftar Isi	2
Latar Belakang	3
Spesifikasi Umum	4
System Mechanics	4
1. About the System	4
2. Main Menu	4
3. Command	4
a. Command Dasar	5
b. SCOREBOARD	5
c. RESET SCOREBOARD	6
d. HISTORY <n>	7
e. RESET HISTORY	8
Konfigurasi Sistem	9
Spesifikasi Game	10
1. RNG	10
2. Diner Dash	10
3. Hangman	10
4. Tower of Hanoi	12
5. Snake on Meteor	13
6. Game tambahan/ Buatan pemain	24
Daftar ADT yang Digunakan	25
1. ADT Stack	25
2. ADT Set & Map	25
3. ADT Linked List	25
Bonus	26
Catatan Tambahan	28

Latar Belakang



“BNMO, robot video game console yang pernah rusak”

BNMO (dibaca: Binomo) adalah sebuah robot video game console yang dimiliki oleh Indra dan Doni. Dua bulan yang lalu, ia mengalami kerusakan dan telah berhasil diperbaiki. Sayangnya, setelah diperbaiki ia justru mendapatkan lebih banyak bug dalam sistemnya. Oleh karena itu, Indra dan Doni mencari programmer lain yang lebih handal untuk ulang memprogram robot video game console kesayangannya. Pada tugas sebelumnya, kalian telah berhasil membuat Indra dan Doni bahagia dengan mengimplementasikan fitur-fitur dasar. Kini, Indra dan Doni ingin melakukan pengembangan lebih lanjut dengan menambahkan fitur serta permainan pada BNMO.

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**

System Mechanics

1. About the System

BNMO merupakan suatu robot game console yang dapat menjalankan permainan. BNMO memiliki beberapa fitur utama, yaitu:

1. Memainkan game
2. Menambahkan game
3. Menghapus game
4. Mengurutkan game yang akan dimainkan
5. Menampilkan game yang telah dimainkan
6. Menampilkan scoreboard game

2. Main Menu

Ketika program pertama kali dijalankan, BNMO akan memperlihatkan main menu yang berisi welcome page dan beberapa menu pilihan yaitu START dan LOAD. Setelah itu, main menu akan menerima input commands yang akan dijelaskan pada bagian berikutnya.

3. 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.

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: **4**

APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD TOWER OF HANOI (YA/TIDAK)? **YA**

Scoreboard berhasil 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.

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

Konfigurasi Sistem

File konfigurasi akan dibaca saat memulai permainan. File ini menyimpan data-data yang disimpan ketika sistem dijalankan sebelumnya.

File konfigurasi memiliki spesifikasi sebagai berikut:

1. File konfigurasi awal

Isi file konfigurasi	Keterangan
5 RNG Diner DASH HANGMAN TOWER OF HANOI SNAKE ON METEOR	Jumlah Permainan

2. File yang disimpan oleh sistem

Contoh file yang disimpan kedalam sistem:

Isi file konfigurasi	Keterangan
6 RNG Diner DASH HANGMAN TOWER OF HANOI SNAKE ON METEOR GAME ASAL 3 HANGMAN TOWER OF HANOI SNAKE ON METEOR 2 BNMO 19 Finn 12 3 Jake 58 Finn 31 Marcelline 30 0	Jumlah permainan Hasil CREATE GAME Jumlah history permainan Jumlah scoreboard game RNG Jumlah scoreboard game Diner DASH Jumlah scoreboard game HANGMAN

0	Jumlah scoreboard game TOWER OF HANOI
1	Jumlah scoreboard game SNAKE ON METEOR
Marshall 77	
0	Jumlah scoreboard game hasil CREATE GAME

Penjelasan file konfigurasi:

File konfigurasi terbagi menjadi 3 bagian, yaitu bagian list permainan, history, dan scoreboard.

1. Baris pertama merupakan sebuah angka X yang menunjukkan jumlah permainan yang dimiliki oleh sistem
X baris berikutnya adalah nama permainan yang dimiliki oleh sistem
2. Baris selanjutnya merupakan sebuah angka Y yang menunjukkan jumlah history permainan yang dimiliki oleh sistem
Y baris berikutnya adalah nama permainan yang terdapat dalam history
3. Baris selanjutnya merupakan sebuah angka Z yang menunjukkan jumlah baris dari scoreboard permainan
Z baris berikutnya adalah nama pemain dan skor yang diperoleh

Spesifikasi Game

1. RNG

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

2. Diner Dash

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

3. Hangman

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

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

ditentukan di *code*. Kemudian, pemain diberikan 10 kesempatan untuk melakukan menebak 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 kreatif 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 tebak: **t**

Berhasil menebak kata MATA! Kamu mendapatkan 4 poin!

Tebakan sebelumnya: -

Kata: _ _ _ _ _

Kesempatan: 8

Masukkan tebak:

Permainan berlanjut hingga kesempatan habis

4. Tower of Hanoi

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

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

- Contoh (tampilan tidak harus ditiru, tampilannya sangat dibebaskan dan boleh kreatif mungkin):

Awal			Akhir		
<pre> * *** ***** ******** ********* ----- A B C </pre>			<pre> * *** ***** ******** ********* ----- A B C </pre>		
<p>TIANG ASAL: A TIANG TUJUAN: B</p>			<p>Kamu berhasil!</p>		
<p>Memindahkan piringan ke B...</p>			<p>Skor didapatkan: 10 Nama: BNMO</p>		

- Contoh jika tiang hanya terisi sebagian.

```

      |
      |
      |
     ***
    *****
    -----

```

5. Snake on Meteor

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

Untuk pemahaman spek tubes, kosakata ini akan digunakan:

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

3. **Ekor:** Bagian terakhir dari *snake* (*hint: tail* pada *linked list*)

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

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

$\langle 0,0 \rangle$				$\langle 4,0 \rangle$
$\langle 0,4 \rangle$				$\langle 4,4 \rangle$

- **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>	<table><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td>2</td><td>1</td><td>H</td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></table>													2	1	H										
		2	1	H																						
Kepala H: <0,0>																										

<div>Badan 1: <0,1> Badan 2: <0,2></div>	<table><tr><td>H</td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></table>	H					1					2														
H																										
1																										
2																										
<div>Kepala H: <1,1> Badan 1: <0,1> Badan 2: <0,0></div>	<table><tr><td>2</td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td>H</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></table>	2					1	H																		
2																										
1	H																									

- **Makanan:** Makanan akan diberikan secara random pada sebuah titik $\langle x,y \rangle$ (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 $\langle x,y \rangle$

Proses pertambahan tail adalah sebagai berikut:

- Secara umum**, ekor *snake* akan bertambah pada titik ordinat yang sama dengan tail, tetapi dengan titik koordinat satu sebelum tailnya (**Apabila tail berada pada titik $\langle x,y \rangle$, maka pertambahan tail akan dilakukan pada titik $\langle x-1,y \rangle$**).
- Apabila kasus a tidak mungkin** (misalkan karena tail berada pada titik $\langle 0,1 \rangle$ dan tidak memungkinkan ada nilai titik $\langle -1,1 \rangle$), maka pertambahan akan dilakukan pada titik ordinat satu sebelum ekor *snake* sekarang, tetapi pada titik koordinat yang

sama (Apabila tail berada pada titik $\langle x,y \rangle$, maka penambahan tail akan dilakukan pada titik $\langle x,y-1 \rangle$)

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

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

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

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

Contoh:

- a. Turn 0: Kepala *snake* berada pada titik $\langle 4,2 \rangle$ (titik H) dengan badannya berada pada $\langle 3,2 \rangle$ (titik 1) dan $\langle 2,2 \rangle$ (titik 2) secara berurutan (maka *snake* memiliki urutan $\langle 4,2 \rangle, \langle 3,2 \rangle, \langle 2,2 \rangle$)

<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 bergerak 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.

6. Game tambahan/ Buatan pemain

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

Daftar ADT yang Digunakan

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

1. ADT Stack

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

2. ADT Set & Map

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

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

3. ADT Linked List

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

Bonus

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

1. Game custom menggunakan ADT Tree

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

2. Penambahan fitur pada game Hangman

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

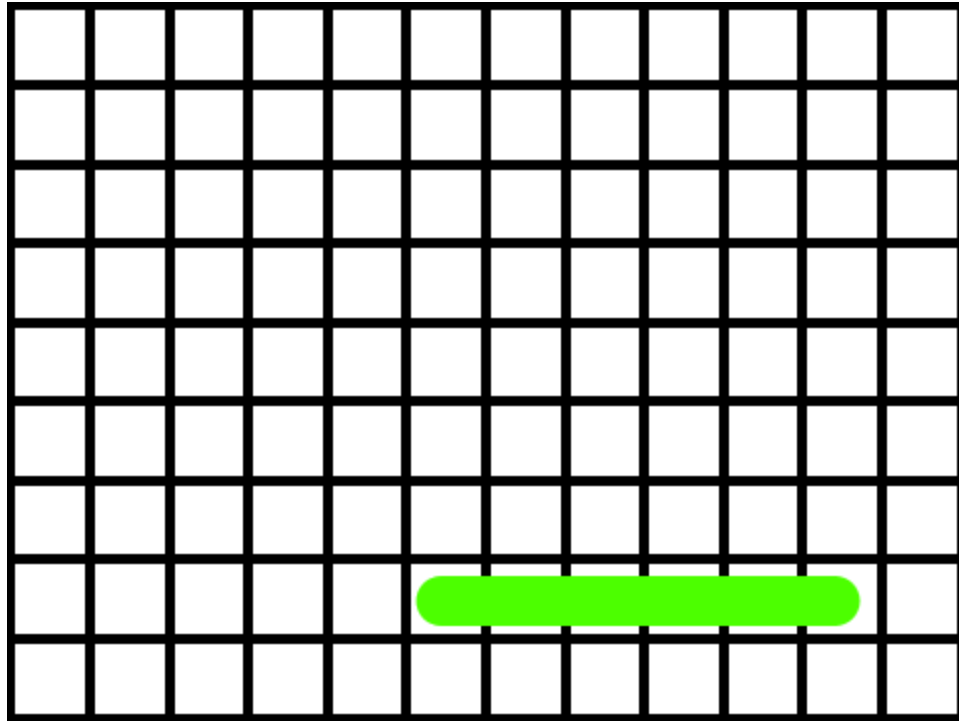
3. Penambahan fitur pada game Tower of Hanoi

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

4. Penambahan fitur pada game Snake on Meteor

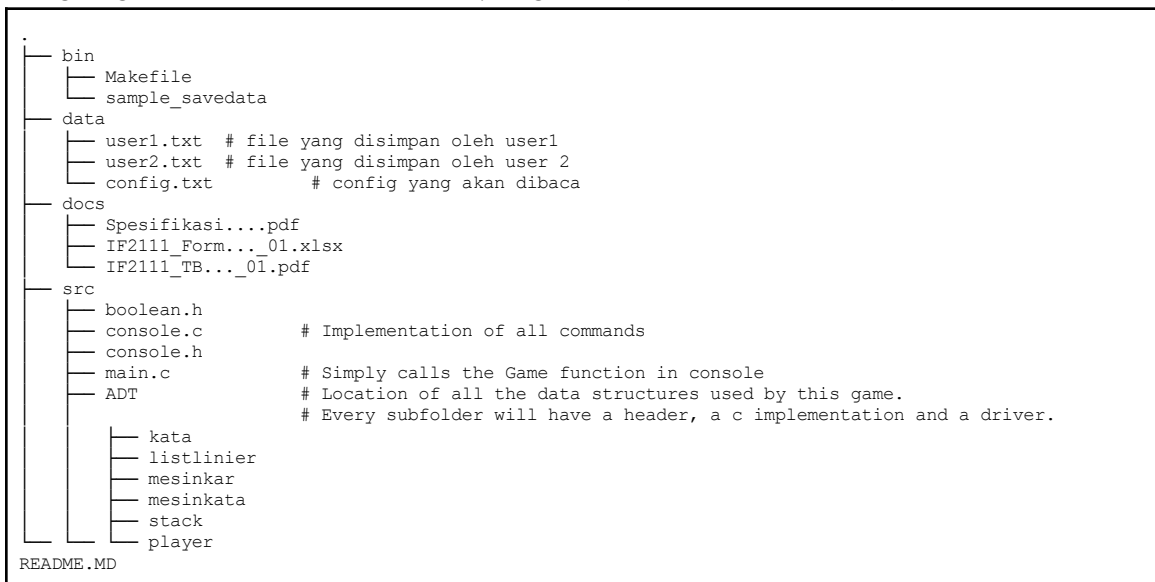
- **Penambahan obstacle.** Ketika Kepala dari *snake* mengenai obstacle, maka permainan berakhir. *Obstacle* muncul di awal permainan dan tidak dapat ditembus oleh *snake*. Selain itu, makanan juga tidak dapat muncul pada titik yang memiliki *obstacle*.

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



Catatan Tambahan

1. Tampilan program boleh dibuat sesuai keinginan kalian, tampilan pada spesifikasi ini hanya merupakan contoh.
2. Diwajibkan untuk membuat driver untuk masing-masing ADT. Driver berisi sebuah main file yang memanggil fungsi/prosedur yang ada di ADT tersebut. Kegunaan driver adalah untuk testing ADT yang sudah dibuat.
3. Sebagai saran, manfaatkan Makefile untuk mempermudah proses kompilasi dan penjalanan program. Bila sulit dalam menggunakan Makefile, bisa diakali dengan menggunakan shell script/batch file.
4. Gunakan Github sebagai version control, lalu invite asisten kalian sebagai collaborator. Pastikan asisten sudah masuk ke dalam repository sebelum asistensi 1.
5. Buat file readme yang minimal mengandung deskripsi singkat program, identitas anggota kelompok dan cara kompilasi program. Readme dapat dibuat dengan menggunakan [markdown](#).
6. Buat struktur program yang serapi mungkin. Jangan buat semuanya pada file yang sama. Contoh struktur program (tidak harus diikuti):



7. Manfaatkan ADT yang sudah kalian buat dalam praktikum semaksimal mungkin.
8. Perhatikan bahwa nilai untuk bonus akan lebih kecil dibandingkan dengan fitur utama. Silakan prioritaskan fitur-fitur yang lebih penting terlebih dahulu.
9. Jika ada yang kurang jelas, silahkan bertanya melalui [FAQ](#). FAQ akan diperiksa setidaknya sekali sehari oleh asisten.