

# **LAPORAN TUGAS BESAR**

## **IF2111 Algoritma dan Struktur Data**

### **BNMO**


Dipersiapkan oleh:

#### **Kelompok 6**

Nicholas Francis Aditjandra	18221005
Sri Laksmi Purwoningtyas	18221009
Farchan Martha Adji Chandra	18221011
Raden Sjora Okalani	18221014
Fariz Putra Hanggara	18221015
Aniqa Fayyaza Akbar	18221020

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	<b>Sekolah Teknik Elektro dan Informatika ITB</b>	<b>Nomor Dokumen</b>		<b>Halaman</b>
		<i>IF2111-TB2-06</i>		32
		<i>Revisi</i>		<i>2 Desember 2022</i>

# Daftar Isi

<b>1 Ringkasan</b>	<b>2</b>
<b>2 Penjelasan Tambahan Spesifikasi Tugas</b>	<b>4</b>
2.1 Fitur Tambahan pada Hangman	4
<b>3 Struktur Data (ADT)</b>	<b>4</b>
3.1 ADT Stack	4
3.2 ADT Set & Map	5
3.3 ADT Linked List	5
<b>4 Program Utama</b>	<b>5</b>
<b>5 Data Test</b>	<b>6</b>
5.1 Data Test 1	6
5.2 Data Test 2	7
5.3 Data Test 3	7
5.4 Data Test 4	8
5.5 Data Test 5	8
5.6 Data Test 6	8
5.7 Data Test 7	9
<b>6 Test Script</b>	<b>9</b>
<b>7 Pembagian Kerja dalam Kelompok</b>	<b>10</b>
<b>8 Lampiran</b>	<b>11</b>
8.1 Deskripsi Tugas Besar 2	11
8.2 Notulen Rapat	26
8.3 Log Activity Anggota Kelompok	26
8.4 Hasil Asistensi	27

# 1 Ringkasan

Tugas Besar 2 meminta mahasiswa untuk melanjutkan Tugas Besar 1, yang meminta mahasiswa untuk membuat BNMO, sebuah *video game console* berbasis *command-line interface* dalam bahasa C. BNMO pada Tugas Besar 2 memiliki fitur utama yang sama disertai dengan dua fitur tambahan, yaitu menampilkan *game* yang telah dimainkan, menghapus daftar *game* yang pernah dimainkan, menampilkan *scoreboard game*, dan menghapus *scoreboard game*. Program ini dibuat dengan menggunakan dan memodifikasi struktur data yang sebelumnya telah dipelajari, seperti ADT Queue, ADT Array Dinamis, ADT Mesin Karakter, ADT Mesin Kata, ADT Stack, ADT Set & Map, dan ADT *Linked List*. Program ini dimulai dengan menampilkan *welcome page* dan meminta input “START” atau “LOAD <file>.txt”.

Setelah program membaca masukan pertama, pemain dapat meng-*input* berbagai *command* yang tersedia, yaitu SAVE, CREATEGAME, LISTGAME, DELETGAME, QUEUEGAME, PLAYGAME, SKIPGAME, QUIT, HELP, SCOREBOARD, RESET SCOREBOARD, HISTORY, dan RESET HISTORY. Berikut penjelasan singkat tentang *command* yang tersedia:

1. START: membaca *file* konfigurasi *default*
2. LOAD: membaca *file* yang dituliskan
3. SAVE: menyimpan *state* game pemain ke dalam suatu *file*
4. CREATEGAME: menambahkan *game* baru pada daftar game
5. LISTGAME: menampilkan daftar *game* yang tersedia
6. DELETGAME: menghapus *game* dari daftar *game*
7. QUEUEGAME: mendaftarkan *game* ke dalam *list*
8. PLAYGAME: memainkan *game*
9. SKIPGAME: melewati permainan
10. QUIT: keluar dari program
11. HELP: menampilkan *command* yang tersedia
12. SCOREBOARD: menampilkan nama dan skor untuk semua *game*
13. RESET SCOREBOARD: menghapus semua informasi *scoreboard*, baik pada setiap permainan, maupun salah satu permainan.
14. HISTORY: melihat daftar permainan yang telah dimainkan
15. RESET HISTORY: menghapus semua *history* permainan yang dimainkan

Apabila pemain memasukkan *input*-an yang tidak sesuai dengan *command* yang ada, program akan menampilkan pesan “Command tidak dikenali, silahkan masukkan command yang valid.” dan setelah itu akan terus meminta *input*-an yang sesuai. Terdapat lima game yang tersedia di program ini, yaitu RNG, Diner Dash, Hangman, Tower of Hanoi, dan Snake on Meteor. Berikut penjelasan singkat mengenai permainan yang tersedia:

1. RNG: permainan tebak angka
2. Diner Dash: permainan pengantaran makan berdasarkan prioritas
3. Hangman: permainan tebak kata
4. Tower of Hanoi: permainan memindahkan piring dengan aturan piringan yang bawah tidak boleh lebih kecil daripada piringan yang ada di atasnya.
5. Snake on Meteor: permainan menggerakkan ular dengan menghindari meteor

Laporan ini berisi penjelasan singkat mengenai persoalan dalam Tugas Besar 2. Selain itu, laporan ini juga memuat penjelasan mengenai spesifikasi fitur, ADT yang digunakan dalam pengerjaan, penjelasan mengenai program utama, *data test*, dan *test script*. Di bagian akhir laporan ini terdapat pembagian kerja kelompok dan berbagai lampiran, di antaranya deskripsi Tugas Besar 2, notulen rapat, *log activity*, dan hasil asistensi.

Di akhir pengerjaan tugas ini, mahasiswa telah berhasil membuat BNMO, sebuah permainan berbasis *command-line interface* dengan berbagai fitur tambahan baru. Mahasiswa telah menerapkan bahasa C dan struktur data yang sebelumnya telah dipelajari di kelas dan praktikum. Selain itu, dengan mengerjakan tugas ini, mahasiswa belajar cara membuat program dengan alur yang *complex*.

## 2 Penjelasan Tambahan Spesifikasi Tugas

### 2.1 Fitur Tambahan pada Hangman

Fitur tambahan yang ada di program kami terdapat pada *game* hangman, yaitu fitur untuk membaca *list* kata dari sebuah file hangman.txt menggunakan mesin kata. Kemudian, *list* kata tersebut akan dipindahkan ke sebuah array dinamis. Selain itu, di *game* hangman terdapat fitur untuk pemain menambahkan kata sendiri yang kemudian akan dimasukkan ke array *list* kata dan hangman.txt.

## 3 Struktur Data (ADT)

### 3.1 ADT Stack

Struktur data stack yang kami gunakan terdiri buffer yang bertipe `char* ElTypeStack` untuk penyimpanan elemen dan `idxTop` yang bertipe integer untuk menyimpan alamat puncak. Elemen dari stack bisa diakses dengan `<namastack>.buffer[<indeks>]`. ADT ini digunakan dalam penyimpanan history, penyimpanan command `printhistory`, dan `resethistory`. Kami memilih stack untuk command history untuk menggambarkan urutan game yang telah dimainkan. Sketsa yang kami gunakan dari ADT ini terdiri dari:

- `void CreateStack(Stack *S)`  
digunakan untuk membuat stack history kosong
- `boolean IsEmptyStack(Stack S)`  
Digunakan untuk mengetahui stack history kosong atau tidak
- `void push(Stack *S, EltypeStack val)`  
Digunakan untuk menambah game yang baru dimainkan ke dalam stack history
- `void pop(Stack *s, EltypeStack *Val)`  
Digunakan untuk menghapus elemen history paling atas dari stack history
- `int lengthstack(Stack s)`  
Digunakan untuk mencari panjang stack history

Selain itu, ADT Stack juga digunakan di program kami di permainan Tower of Hanoi untuk menggambarkan perpindahan piring dari satu tiang ke tiang lainnya. Berbeda dengan stack yang dipakai untuk history, elemen pada stack untuk Tower of Hanoi bertipe int Typehanoi.

### 3.2 ADT Set & Map

ADT Set & Map digunakan untuk penyimpanan skor dan nama user kedalam scoreboard dan juga untuk mengimplementasikan fungsi savescoreboard dan printscoreboard. Selain itu, ADT Map juga digunakan di permainan Snake on Meteor untuk *grid* permainannya. Di dalam ADT Map terdapat set untuk memastikan bahwa tidak ada nama yang sama di dalam *scoreboard*. Sketsa yang kami gunakan dalam ADT ini terdiri dari:

- void CreateEmptyMap (Map \*M)  
Digunakan untuk membuat map game yang kosong dan menghapus scoreboard menjadi kosong
- void IsEmptyMap (Map M)  
Digunakan untuk mengecek apakah map game kosong atau tidak
- void InsertMap (Map \*M, keytype k, valuetype v)  
Memasukkan username dan skor kedalam mapgame

### 3.3 ADT Linked List

Variasi dari linked list yang kami pakai adalah list double pointer. Di ADT ini terdapat tiga struktur data yaitu point, list, dan elmtlist. Point terdiri dari integer x dan y. Elmtlist terdiri dari integer info, addressLDP next, addressLDP prev, dan point coor. List terdiri dari addressLDP first dan addressLDP last. Kami memilih ADT *linked list* untuk menggambarkan bagian *tubuh* snake yang saling terhubung di permainan Snake on Meteor. Beberapa sketsa yang kami gunakan dalam ADT ini terdiri dari:

- boolean IsEmptyListSNAKE (ListSNAKE L)  
Digunakan untuk mengecek apakah snake memiliki linked list sebagai bagian tubuh.
- void CreateEmptyListSNAKE (ListSNAKE \*L)  
Digunakan untuk membuat bagian tubuh yang kosong
- void InsertTailSNAKE (ListSNAKE \*L, addressSNAKE P)  
Digunakan untuk menambahkan tail pada list snake
- void DelTailSNAKE (ListSNAKE \*L, addressSNAKE \*P)  
Digunakan untuk menghapus tail pada list snake

## 4 Program Utama

Program utama dimulai dengan menampilkan *interface* BNMO dan pemain akan memilih antara *command* START atau *command* LOAD. Saat pemain memasukkan *command* START sistem akan membaca *file* konfigurasi dan menampilkan bahwa sistem BNMO berhasil dijalankan. Sedangkan saat pemain memasukkan *command* LOAD, sistem akan membaca suatu *file* yang di-*input* oleh pemain.

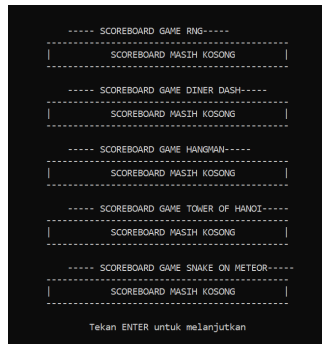
Setelah sistem berhasil membaca konfigurasi, pemain dapat memasukkan *command-command* yang tersedia pada sistem. Pertama, CREATE GAME yang berfungsi menambahkan *game* yang belum ada pada *list game* sesuai masukkan dari pemain. Kedua, LIST GAME yang akan menampilkan daftar *game-game* yang tersedia dari *file* konfigurasi yang telah dibaca sebelumnya. Ketiga, DELETE GAME yang berfungsi menghapus *game* pada *list* sebelumnya sesuai nomor *input*-an dari pemain. Keempat, QUEUE GAME untuk menambahkan *game* ke antrean sebelum permainan tersebut dapat dimainkan. Kelima, SKIP GAME. Saat pemain belum menambahkan *game* ke antrean, saat memasukkan *command* SKIP GAME akan menampilkan bahwa antrean kosong. Saat sudah terdapat daftar *game* pada antrean, sistem akan meminta pemain untuk memberikan masukan berupa jumlah *game* yang ingin dilewati. Saat permainan berhasil dilewati, akan menampilkan sisa *game* pada daftar antrean. *Command* keenam adalah PLAY GAME. Sebelum memasukkan *command* PLAY GAME, pemain sebelumnya harus memasukkan *game-game* ke dalam antrian terlebih dahulu melalui *command* QUEUE GAME. Selanjutnya sistem akan menampilkan *list* permainan pada antrean dan memulai permainan sesuai dengan urutan pada antrean *game*. *Command* ketujuh adalah HELP. *Command* HELP akan menampilkan *command-command* yang tersedia pada sistem. Kedelapan, SAVE yang digunakan untuk menyimpan data permainan pemain ke dalam suatu *file*. Terakhir, QUIT merupakan *command* yang ketika dimasukkan oleh pemain akan membuatnya keluar dari program. Saat pemain memasukkan *command* yang tidak sesuai, akan menampilkan bahwa *command* tidak dikenali dan pemain diminta untuk memasukkan *command* baru.

Di program utama algoritma ini terdapat empat *command* baru yang dapat dimasukkan oleh pemain. Pertama, *command* SCOREBOARD yang menampilkan nama dan skor untuk semua *game*. Kedua, *command* RESET SCOREBOARD yang merupakan *command* untuk menghapus informasi terhadap scoreboard permainan. Reset dapat dilakukan untuk menghapus semua informasi pada setiap permainan maupun memilih salah satu permainan untuk dihapus informasinya. *Command* HISTORY <n> digunakan untuk menampilkan permainan yang telah ditampilkan, dengan <n> adalah jumlah permainan yang telah dimainkan yang ingin ditampilkan. Terakhir, terdapat *command* RESET HISTORY yang digunakan untuk menghapus semua *history* permainan yang telah dimainkan.

## 5 Data Test

### 5.1 Data Test 1

Test ini dilakukan untuk memastikan input SCOREBOARD pada main menu bisa dijalankan.



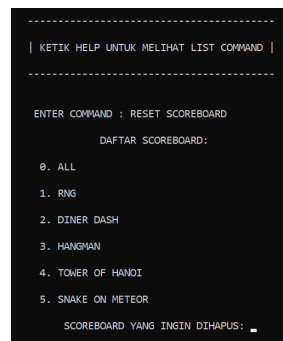
Gambar 5.1.1 Tampilan scoreboard di awal program



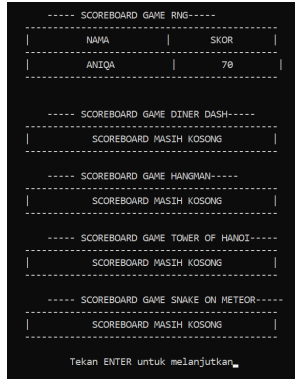
Gambar 5.1.2 Tampilan scoreboard ketika sudah ada game yang dimainkan

### 5.2 Data Test 2

Test ini dilakukan untuk memastikan input RESET SCOREBOARD pada main menu bisa dijalankan.



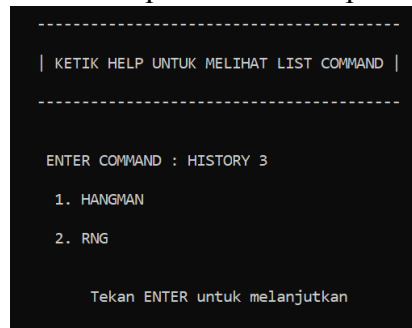
Gambar 5.2.1 Tampilan ketika menjalankan *command* RESET SCOREBOARD



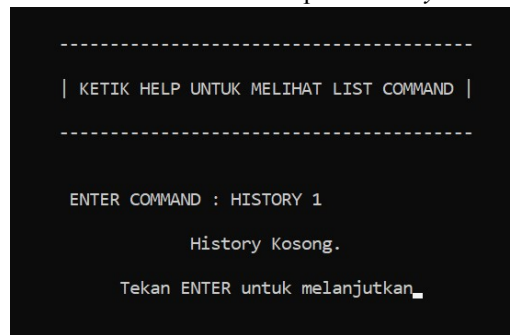
**Gambar 5.2.2** Tampilan setelah *scoreboard* hangman di-reset

### 5.3 Data Test 3

Test ini dilakukan untuk memastikan input HISTORY pada main menu bisa dijalankan.



**Gambar 5.3.1** Tampilan *history*

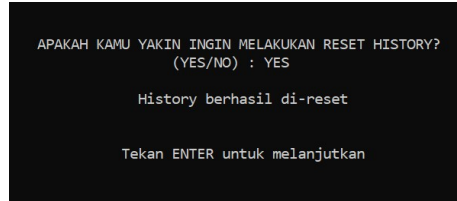


**Gambar 5.3.2** Tampilan *history* setelah di-reset

### 5.4 Data Test 4

Test ini dilakukan untuk memastikan input RESET HISTORY pada main menu bisa dijalankan.

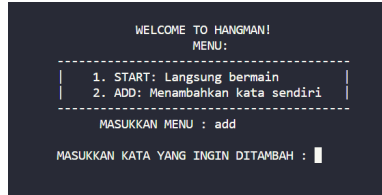




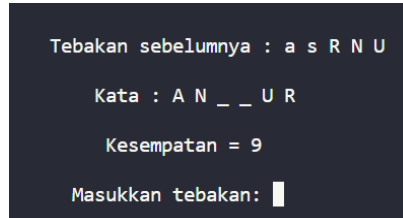
**Gambar 5.4.1** Tampilan *history* setelah di-reset

## 5.5 Data Test 5

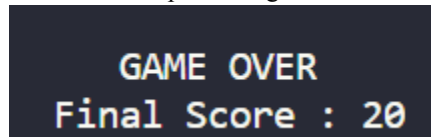
Test ini dilakukan untuk memastikan game HANGMAN bisa dijalankan.



**Gambar 5.5.1** Tampilan hangman saat pertama mulai dan memilih menu *add*



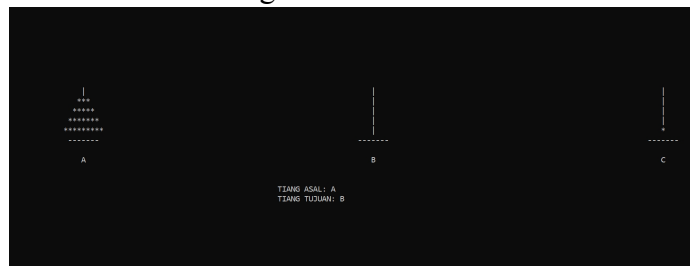
**Gambar 5.5.2** Tampilan hangman ketika bermain



**Gambar 5.5.3** Tampilan hangman setelah *game* selesai

## 5.6 Data Test 6

Test ini dilakukan untuk memastikan game TOWER OF HANOI bisa dijalankan.



**Gambar 5.6.1** Tampilan Tower of Hanoi ketika ingin memindahkan piringan



**Gambar 5.6.2** Tampilan Tower of Hanoi ketika berhasil memindahkan piringan

## 5.7 Data Test 7

Test ini dilakukan untuk memastikan game SNAKE ON METEOR bisa dijalankan.



**Gambar 5.7.1** Tampilan Snake on Meteor ketika dimainkan

## 6 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 program dapat menjalankan command SCOREBOARD	Memasukkan command SCOREBOARD ketika program meminta input di main menu sebelum dan setelah memainkan game	Data Test 1	Menampilkan Scoreboard dari setiap game yang ada	Sesuai dengan yang diharapkan
2	Fitur RESET SCOREBOARD	Memeriksa apakah program dapat menjalankan command RESET SCOREBOARD	Memasukkan command RESET SCOREBOARD ketika program meminta input di main menu dan memasukkan data scoreboard yang ingin di-reset	Data Test 2	Me-reset scoreboard yang diinginkan	Sesuai dengan yang diharapkan
3	Fitur HISTORY	Memeriksa apakah program dapat menjalankan	Memasukkan command HISTORY ketika program meminta input di main menu	Data Test 3	Menampilkan history permainan yang pernah dimainkan	Sesuai dengan yang diharapkan

		command HISTORY	dan memasukkan jumlah permainan yang telah dimainkan untuk ditampilkan		dalam jumlah yang diinginkan	
4	Fitur RESET HISTORY	Memeriksa apakah program dapat menjalankan command RESET HISTORY	Memasukkan command RESET HISTORY ketika program meminta input di main menu	Data Test 4	Menghapus daftar game yang pernah dimainkan	Sesuai dengan yang diharapkan
5	Fitur HANGMAN	Memeriksa apakah program dapat memainkan game Hangman	Menginput kata dengan fitur <i>add</i> dan bermain hangman	Data Test 5	Kata bertambah di dictionary dan skor terlihat setelah pemain kehabisan kesempatan	Sesuai dengan yang diharapkan
6	Fitur TOWER OF HANOI	Memeriksa apakah program dapat memainkan game Tower of Hanoi	Memasukkan input huruf tiang asal dan tiang tujuan	Data Test 6	Memindahkan piringan dari tiang asal ke tiang tujuan	Sesuai dengan yang diharapkan
7	Fitur SNAKE ON METEOR	Memeriksa apakah program dapat memainkan game Snake on Meteor	Memasukkan input command huruf 'a', 'w', 's', atau 'd'	Data Test 7	Head dari snake berpindah sesuai command yang diminta	Sesuai dengan yang diharapkan

## 7 Pembagian Kerja dalam Kelompok

No.	Fitur	NIM Coder	NIM Tester
1.	Main program	18221011	18221020 18221014
2.	Scoreboard	18221014	18221020
3.	Reset Scoreboard	18221011	18221014
4.	History	18221009	18221011

5.	Reset History	18221011	18221005
6.	Hangman	18221020	18221011 18221014
7.	Tower of Hanoi	18221015	18221005 18221009
8.	Snake On Meteor	18221005	18221011 18221014

## 8 Lampiran

### 8.1 Deskripsi Tugas Besar 2

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

#### System Mechanics

##### 1. About the System

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

- Memainkan game
- Menambahkan game
- Menghapus game
- Mengurutkan game yang akan dimainkan
- Menampilkan game yang telah dimainkan
- 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 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.

### 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	Jumlah permainan      Hasil CREATE GAME Jumlah history permainan   Jumlah scoreboard game RNG   Jumlah scoreboard game Diner DASH

Finn 31	
Marcelline 30	
0	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 di laporan Tugas Besar 1.
2. Diner Dash  
Dijelaskan di laporan 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 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 terbak 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, BNMO melihat panduannya di The Enchiridion. Setelah itu, BNMO ingin melakukan modifikasi permainan ini:

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

#### 5. Snake on Meteor

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



Untuk pemahaman spek tubes, kosakata ini akan digunakan:

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

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

- Dimensi Peta: Dimensi peta adalah 5x5 unit dengan  $\langle 0,0 \rangle$  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></table>																	2	1	H
		2	1	H																

	<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></table>																									
Kepala H: <0,0> Badan 1: <0,1> Badan 2: <0,2>	<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																										
Kepala H: <1,1> Badan 1: <0,1> Badan 2: <0,0>	<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  $\langle x-1,y \rangle$ ).

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

Contoh:

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

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

		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  $\langle 3,3 \rangle$  (titik H),  $\langle 2,3 \rangle$  (titik 1),  $\langle 1,3 \rangle$  (titik 2),  $\langle 0,3 \rangle$  (titik 3) dan meteor menyerang peta pada titik 2. Maka bagian badan  $\langle 1,3 \rangle$  akan dihapus dan sekarang anggota tubuh berupa  $\langle 3,3 \rangle, \langle 2,3 \rangle, \langle 0,3 \rangle$

Sebelum terkena meteor	<table><tr><td>&lt;0,0&gt;</td><td></td><td></td><td></td><td>&lt;4,0&gt;</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>3</td><td>2</td><td>1</td><td>H</td><td></td></tr><tr><td>&lt;0,4&gt;</td><td></td><td></td><td></td><td>&lt;4,4&gt;</td></tr></table>	<0,0>				<4,0>											3	2	1	H		<0,4>				<4,4>
<0,0>				<4,0>																						
3	2	1	H																							
<0,4>				<4,4>																						
Saat terkena meteor	<table><tr><td>&lt;0,0&gt;</td><td></td><td></td><td></td><td>&lt;4,0&gt;</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>3</td><td>m</td><td>1</td><td>H</td><td></td></tr><tr><td>&lt;0,4&gt;</td><td></td><td></td><td></td><td>&lt;4,4&gt;</td></tr></table>	<0,0>				<4,0>											3	m	1	H		<0,4>				<4,4>
<0,0>				<4,0>																						
3	m	1	H																							
<0,4>				<4,4>																						
Setelah terkena meteor	<table><tr><td>&lt;0,0&gt;</td><td></td><td></td><td></td><td>&lt;4,0&gt;</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>2</td><td></td><td>1</td><td>H</td><td></td></tr><tr><td>&lt;0,4&gt;</td><td></td><td></td><td></td><td>&lt;4,4&gt;</td></tr></table>	<0,0>				<4,0>											2		1	H		<0,4>				<4,4>
<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  $\langle 4,3 \rangle$  dan badannya  $\langle 3,3 \rangle, \langle 2,3 \rangle$  (posisi *snake* secara berurutan:  $\langle 4,3 \rangle, \langle 3,3 \rangle, \langle 2,3 \rangle$ ). 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  $\langle 0,0 \rangle$  dan titik maksimal dari peta adalah  $\langle 4,4 \rangle$ . Apabila kepala berada pada titik  $\langle 0,0 \rangle$  dan dilakukan *input* 'a', maka pada putaran berikutnya kepala tidak harus berada pada titik  $\langle 4,0 \rangle$

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

## 8.2 Notulen Rapat

Rapat pertama (18 November 2022)

- Pembagian Tugas
- Mulai mengerjakan tugas secara mandiri

Rapat kedua (1 Desember 2022)

- Menyempurnakan Snake on Meteor
- Melanjutkan pengerjaan laporan

Rapat ketiga (2 Desember 2022)

- Menyelesaikan laporan
- Merapikan source code
- Mengupload source code ke github

### 8.3 Log Activity Anggota Kelompok

No.	Tanggal	NIM	Nama	Aktivitas
1.	18/11/2022	18221005 18221009 18221011 18221014 18221015 18221020	Nicholas Francis Sri Laksmi Farchan Martha Sjora Okalani Fariz Putra Aniqa Fayyaza	Pembagian tugas
2.	24/11/2022	18221005 18221009 18221011 18221014 18221015 18221020	Nicholas Francis Sri Laksmi Farchan Martha Sjora Okalani Fariz Putra Aniqa Fayyaza	Asistensi 1
3.	24/11/2022 - 1/12/2022	18221005 18221009 18221011 18221014 18221015 18221020	Nicholas Francis Sri Laksmi Farchan Martha Sjora Okalani Fariz Putra Aniqa Fayyaza	Pengerjaan tugas mandiri sesuai dengan pembagian tugas sebelumnya
4.	1/12/2022	18221005 18221009 18221011 18221014 18221015 18221020	Nicholas Francis Sri Laksmi Farchan Martha Sjora Okalani Fariz Putra Aniqa Fayyaza	Asistensi 2
5.	1/12/2022 - 2/12/2022	18221005 18221009 18221011 18221014 18221015 18221020	Nicholas Francis Sri Laksmi Farchan Martha Sjora Okalani Fariz Putra Aniqa Fayyaza	Penyelesaian laporan

### 8.4 Hasil Asistensi

**Form Asistensi Tugas Besar  
IF2110/Algoritma dan Struktur Data**



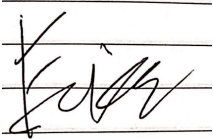

STEI- ITB	IF2111-TB2-06	Halaman 27 dari 32 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		




No. Kelompok/Kelas : 06/K03  
Nama Kelompok : Three Point Six  
Anggota Kelompok (Nama/NIM) :  
1. Nicholas Francis Aditjandra / 18221005  
2. Sri Laksmi Purwoningtyas / 18221009  
3. Farchan Martha Adji Chandra / 18221011  
4. Raden Sjora Okalani / 18221014  
5. Fariz Putra Hanggara / 18221015  
6. Anika Fayyaza Akbar / 18221020

Asisten Pembimbing : Kadek Surya Mahardika



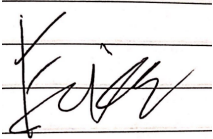

---




# Asistensi I

<b>Tanggal : 24 November 2022</b>	<b>Catatan Asistensi:</b>
<b>Tempat :</b> <a href="https://itb-ac-id.zoom.us/j/2113314820?pwd=VEIyM0k4bkSyUmxFOXcWd2hFTWc3dz09">https://itb-ac-id.zoom.us/j/2113314820?pwd=VEIyM0k4bkSyUmxFOXcWd2hFTWc3dz09</a>	<ol style="list-style-type: none"> <li>1. Kalo hangman nebak huruf yang sebelumnya udah pernah ditebak, kesempatannya berkurang gak? Harusnya kesempatannya gak berkurang</li> </ol>
<b>Kehadiran Anggota Kelompok:</b> <div style="text-align: center;"> No  NIM  Tanda tangan </div> <div style="text-align: center;"> 1  18221020   </div> <div style="text-align: center;"> 2  18221014   </div> <div style="text-align: center;"> 3  18221015   </div> <div style="text-align: center;"> 4  18221005   </div> <div style="text-align: center;"> 5  18221009 </div>	<ol style="list-style-type: none"> <li>2. Isi file konfigurasi kan ada jumlah permainan dan baris setelah jumlah history ada jumlah scoreboard, itu urutannya random atau gimana? Ngikutin sesuai yang di jumlah permainan</li> <li>3. Daftar ADT yang digunakan ada set dan map dan tiap game dibedain, jadi klo dia buat game baru, dia buat set baru buat game itu? Iya</li> </ol>

  <p>6 18221011</p> 	
	<p><b>Tanda Tangan Asisten:</b></p>  <p><b>Kadek Surya Mahardika</b> 13519165</p>

## Asistensi II

<b>Tanggal : 1 Desember 2022</b>	<b>Catatan Asistensi:</b>
<b>Tempat :</b> <a href="https://itb-ac-id.zoom.us/j/2113314820?pwd=VEIyM0k4bkSyUmxFTWc3dz09">https://itb-ac-id.zoom.us/j/2113314820?pwd=VEIyM0k4bkSyUmxFTWc3dz09</a>	<ol style="list-style-type: none"> <li>1. Tubesnya sudah 70% done, sisa Snake on Meteor</li> </ol>
<b>Kehadiran Anggota Kelompok:</b> <p>No NIM Tanda tangan</p> <p>1 18221020</p>  <p>2 18221014</p>  <p>3 18221015</p>  <p>4 18221005</p>  <p>5 18221009</p>	<ol style="list-style-type: none"> <li>2. Laporan masih <i>on progress</i></li> <li>3. Q: Untuk demo, fokus mengdemokan command Tubes 2 atau tubes 1 juga?</li> <li>4. A: Tubes 2 aja</li> </ol>

  <p>6 18221011</p> 	
	<p><b>Tanda Tangan Asisten:</b></p>   <p><b>Kadek Surya Mahardika</b> <b>13519165</b></p>