

# **LAPORAN TUGAS BESAR**

## **IF2111 Algoritma dan Struktur Data**


### **BNMO Game Console**

Dipersiapkan oleh:

Kelompok 06 IF2111 K01

18221065	Josua Adriel Sinabutar
18221121	Rozan Ghosani
18221123	Abraham Megantoro Samudra
18221143	Lie, Kevin Sebastian S. T.
18221157	Cathleen Laretta

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--01-06</i>		<i>34</i>
		<i>Revisi</i>	<i>1</i>	<i>18 November 2022</i>

# Daftar Isi

<b>1 Ringkasan</b>	<b>2</b>
<b>2 Penjelasan Tambahan Spesifikasi Tugas</b>	<b>3</b>
2.1 Permainan The Glass Game	3
2.2 Fitur Tambahan pada Permainan Hangman	4
2.3 Fitur Tambahan pada Permainan Tower of Hanoi	4
2.4 Fitur Tambahan pada Permainan Snake on Meteor	4
<b>3 Struktur Data (ADT)</b>	<b>5</b>
3.1 ADT stack	5
3.2 ADT set	6
3.3 ADT map	7
3.4 ADT listlinier	7
3.5 ADT listsom	8
3.6 ADT listrek	9
3.7 ADT binTree	10
<b>4 Program Utama</b>	<b>11</b>
<b>5 Data Test</b>	<b>13</b>
5.1 HISTORY <n>	13
5.2 RESET HISTORY	13
5.3 SCOREBOARD	14
5.4 RESET SCOREBOARD	15
5.5 HANGMAN	16
5.6 TOWER OF HANOI	18
5.7 SNAKE ON METEOR	19
5.8 THE GLASS GAME	21
<b>6 Test Script</b>	<b>23</b>
<b>7 Pembagian Kerja dalam Kelompok</b>	<b>26</b>
<b>8 Lampiran</b>	<b>27</b>
8.1 Deskripsi Tugas Besar 2	27
8.2 Notulen Rapat	28
8.3 Log Activity Anggota Kelompok	31

# 1 Ringkasan

Indra dan Doni memiliki sebuah robot game console yang diberi nama BNMO. Pada tugas besar sebelumnya, BNMO yang sempat rusak berhasil diprogram ulang untuk diperbaiki bug yang ada di dalamnya. Indra dan Doni menjadi lebih bahagia dengan fitur dasar yang sudah diimplementasikan pada BNMO sehingga pada kesempatan kali ini, mereka ingin melakukan pengembangan lebih lanjut dengan menambahkan fitur serta permainan pada BNMO.

Sebagai robot game console, BNMO memiliki peran untuk menjalankan permainan yang terdapat di dalamnya. Beberapa fitur dasar yang dapat dijalankan oleh BNMO sudah diimplementasikan pada tugas sebelumnya, antara lain : memainkan game, menambahkan game, menghapus game, dan mengurutkan game yang akan dimainkan. Kini, Indra dan Doni ingin supaya BNMO ditambah beberapa fitur tambahan, seperti : menampilkan game yang telah dimainkan (history) dan menampilkan scoreboard game setiap pemainnya.

Dari tugas besar ini, dapat disimpulkan bahwa persoalan yang dimiliki oleh Indra dan Doni dapat diselesaikan dengan menggunakan ilmu pelajaran yang didapat dari mata kuliah IF2111 Algoritma dan Struktur Data.

## 2 Penjelasan Tambahan Spesifikasi Tugas

### 2.1 Permainan The Glass Game

Permainan The Glass Game merupakan permainan yang diadaptasi dari film *Squid Game*. Pada film tersebut terdapat sebuah permainan yang mengharuskan pemainnya untuk menebak langkah yang harus diambil untuk melewati jurang. Langkah yang mereka ambil merupakan langkah ke kiri atau ke kanan sebuah jembatan kaca. Langkah yang mereka pilih harus benar karena salah satu dari opsi langkah yang ada merupakan sebuah jebakan yang akan membuat pemainnya kalah. Permainan ini diadaptasi dengan menggunakan tipe data binary tree yang memanfaatkan list rekursif.

Fitur ini memiliki beberapa primitif.

1. `generateGlass(BinTree *Glass, int n)` yang digunakan untuk membuat jalur permainan menggunakan binary tree. Jalur yang dibuat merupakan pemilihan secara acak jalur ke kiri atau ke kanan sebanyak  $n$  kali. Primitif ini akan mengembalikan sebuah binary tree yang tiap nodenya unary ke kiri/kanan sebanyak  $n$  buah node.
2. `printCharacter(Word leftOrRight, boolean isCracked)` digunakan untuk mencetak visual permainan ke layar. Prosedur ini menggunakan parameter kata `leftOrRight` untuk menentukan posisi pemain sekarang dan `isCracked` untuk menampilkan variasi kondisi pemain yang terjatuh karena memilih lantai yang salah atau bukan.

Fitur ini dijalankan dengan memanggil prosedur `playTheGlassGame()` untuk menjalankan program. Program akan dimulai pada stage pertama dimana pemain akan diberikan  $(2n + 1)$  buah nyawa dengan  $n$  merupakan stage pemain saat ini. Pemain akan diminta untuk menebak lantai mana yang harus dipijak sebanyak  $(3n)$  lantai dengan  $n$  merupakan stage pemain saat ini. Pemain harus menebak kiri/kanan secara benar berturut-turut. Jawaban tebakan akan dibuat secara random. Pemain yang salah menebak akan terjatuh sehingga nyawanya berkurang. Jika pemain terjatuh, maka pemain harus kembali mengulang ke awal stage dan mengulang kembali urutan

tebakan yang benar seperti sebelumnya. Apabila semua nyawa pemain habis, maka permainan akan berakhir. Pemain yang berhasil menebak semua pijakan yang benar akan lanjut ke stage berikutnya dengan jumlah tebakkan yang lebih banyak. Skor pemain akan dihitung berdasarkan jarak terjauh pemain berhasil menebak jalur pijakan.

## **2.2 Fitur Tambahan pada Permainan Hangman**

Pada permainan Hangman, terdapat fitur tambahan yang memungkinkan pengguna untuk memasukkan kata yang belum tersedia ke dalam *dictionary*. Kata yang dimasukkan pengguna langsung dapat dimunculkan sebagai tebakkan ketika pengguna memilih untuk lanjut bermain. Fitur ini dijalankan dengan pertama membaca masukkan kata dari pengguna yang terdiri dari satu kata saja. Setelah itu, program akan memeriksa apakah kata yang dimasukkan pengguna sudah terdapat pada *dictionary* dan apakah kata yang dimasukkan user terdiri dari satu kata saja serta dituliskan dalam huruf kapital semua. Jika kata sudah terdaftar dalam *dictionary*, maka program akan mengirimkan pesan “Gagal menambahkan Kata! Kata sudah terdapat di dalam dictionary!” dan kata tersebut tidak jadi dimasukkan. Selain itu, jika pengguna memasukkan lebih dari satu kata, maka program akan mengirimkan pesan “INPUT TIDAK VALID! MASUKKAN NAMA BUAH YANG TERDIRI DARI SATU KATA!”. Jika kata yang dimasukkan pengguna valid, program akan mengirimkan pesan “Kata berhasil ditambahkan” dan kata tersebut akan dituliskan ke dalam file *hangman.txt*

Selain fitur menambahkan kata, Hangman juga menerapkan fitur yang membaca list kata atau *dictionary* dari file *hangman.txt*. Fitur ini dijalankan dengan pertama membaca file *hangman.txt* yang konfigurasinya dibuat seperti konfigurasi file pada Tugas Besar 1, lalu memasukkan isi dari file *hangman.txt* ke dalam array sebanyak jumlah kata dalam file tersebut.

## **2.3 Fitur Tambahan pada Permainan Tower of Hanoi**

Fitur tambahan pada permainan Tower of Hanoi memungkinkan pengguna untuk mengubah jumlah piringan yang ingin dimainkan. Pengguna pertama harus memasukkan perintah “CHANGE” untuk mengubah jumlah piringan lalu memasukkan jumlah piringan sebanyak yang diinginkan selama jumlah yang dimasukkan lebih besar dari 0. Jika masukkan valid, maka program akan mengubah jumlah piringan dari yang awalnya 5 (*default*) menjadi sebanyak jumlah piringan yang dimasukkan pengguna. Penghitungan skor juga disesuaikan dengan langkah optimal dari banyaknya jumlah piring dengan rumus  $(2^n - 1)$  dengan  $n$  adalah jumlah piringan yang diinginkan pengguna. Skor maksimal yang bisa didapatkan pemain akan berubah sebanyak dua kali jumlah piringan yang dimainkan.

Selain itu, melihat banyaknya langkah yang mungkin terjadi ketika pengguna memasukkan jumlah piringan yang banyak, maka programmer menambahkan fitur untuk menyerah dan keluar dari permainan ketika permainan sedang berjalan, pengguna dapat mengetikkan “Q” untuk menyerah saat bermain.

## **2.4 Fitur Tambahan pada Permainan Snake on Meteor**

Fitur tambahan pada permainan Snake on Meteor memungkinkan pengguna untuk melewati dinding pembatas permainan dan menembus pada bagian yang berlainan. Dengan

adanya fitur tambahan ini, pengguna tidak akan mati ketika kepala ular melewati dinding pembatas permainan.

Fitur lain yang ditambahkan pada permainan ini adalah adanya rintangan berupa *obstacle* yang tidak bisa dilalui oleh ular. Rintangan ini dibuat secara acak di awal permainan dan tidak akan berubah posisinya hingga permainan berakhir. Ketika kepala ular menabrak rintangan ini, maka ular akan mati dan permainan akan berakhir dengan skor yang didapatkan merupakan dua kali jumlah tubuh ular (tanpa kepala).

### 3 Struktur Data (ADT)

#### 3.1 ADT stack

Struktur data pertama yang digunakan untuk menyelesaikan persoalan mengenai fitur tambahan adalah struktur data Stack. Stack merupakan sederetan elemen yang diimplementasikan dalam bentuk list linier yang dikenali berdasarkan elemen puncaknya (TOP). Karena pada Stack hanya diperbolehkan operasi pada address TOP-nya, maka terdapat beberapa aturan penyisipan dan penghapusan yang harus dipenuhi, seperti :

1. Penyisipan selalu dilakukan di atas (TOP).
2. Penghapusan selalu dilakukan pada elemen puncak (TOP).

Dengan aturan demikian, Stack yang elemennya terdiri atas infotype bertipe Word akan diimplementasikan dalam bentuk ADT stack dengan sketsa struktur data sebagai berikut :

- Memiliki tipe data Stack yang terdiri dari memori tempat penyimpanan elemen infotype bertipe Word sebanyak 100 elemen, serta sebuah integer TOP yang menyatakan alamat/indeks dari elemen puncak Stack. Stack merupakan sebuah “tumpukan” yang mengikuti aturan LIFO (Last In First Out) sehingga operasi penambahan dan pengurangan hanya dapat dilakukan di salah satu “ujung” list.
- Memiliki beberapa fungsi/prosedur yang dapat digunakan untuk melakukan operasi terhadap elemen di dalam Stack, yaitu :
  1. CreateEmptyStack(Stack \*S) yang berfungsi untuk membuat Stack kosong. Stack disebut kosong jika indeks dari elemen puncaknya (TOP) bernilai Undef (-1).
  2. IsEmptyStack(Stack S) yang akan mengembalikan boolean bernilai true/false untuk memeriksa apakah stack pada parameter merupakan stack kosong atau bukan.
  3. IsFullStack(Stack S) yang akan mengembalikan boolean bernilai true/false untuk memeriksa apakah stack pada parameter sudah penuh atau belum. Stack disebut penuh jika indeks elemen puncaknya sudah menyentuh (MaxEl-1).
  4. NbElmtStack(Stack S) yang akan mengembalikan integer yang menyatakan banyaknya elemen dalam stack.
  5. Push(Stack \*S, infotype X) yang berfungsi untuk menambahkan X sebagai elemen puncak dari Stack. Infotype X akan menjadi elemen puncak yang baru dari Stack.

6. Pop(Stack \*S, infotype \*X) yang berfungsi untuk menghapus elemen puncak pada Stack. Nilai dari TOP akan disimpan pada X, kemudian indeks TOP akan “turun” ke elemen di bawah TOP sebelumnya.
- Persoalan yang diselesaikan menggunakan ADT ini adalah persoalan untuk merepresentasikan urutan dari permainan yang telah dimainkan oleh user, dengan TOP sebagai permainan yang baru saja dimainkan. ADT ini juga digunakan sebagai konsep dasar dari game Tower of Hanoi.
  - Alasan mengapa ADT ini digunakan adalah karena konsep yang dimiliki oleh struktur data Stack. Karena elemennya tersusun secara LIFO (Last In First Out), maka hal tersebut dapat mempermudah program untuk menyimpan urutan permainan yang sudah dimainkan oleh user.

### 3.2 ADT set

Struktur data yang digunakan selanjutnya adalah struktur data Set. Set merupakan sekumpulan elemen bertipe sama dan bersifat unik. Pada Set, elemennya tidak berurut sehingga tidak ada istilah elemen "next" dan "previous". Struktur data Set diimplementasikan dalam bentuk ADT set dengan sketsa struktur data yang hampir sama seperti ADT stack, yaitu :

- Memiliki tipe data Set yang terdiri dari memori tempat penyimpanan elemen infotype bertipe Word sebanyak 100 elemen, serta sebuah integer Count yang menyatakan jumlah dari elemen yang terdapat pada Set.
- Memiliki beberapa fungsi/prosedur yang dapat digunakan untuk melakukan operasi terhadap elemen di dalam Set, seperti CreateEmptySet, IsEmptySet, dan IsFullSet yang sama seperti ADT stack. Namun, terdapat beberapa perbedaan pada operasi lain :
  1. IsMemberSet(Set S) yang akan mengembalikan boolean bernilai true/false untuk memeriksa apakah sebuah elemen merupakan anggota dari Set/bukan.
  2. InsertSet(Set \*S, infotype Elmt) yang akan mengembalikan boolean bernilai 0/1 untuk menyatakan apakah sebuah Elmt sudah berada pada Set atau belum. Apabila Elmt sudah berada pada Set, maka prosedur tidak akan menambahkan Elmt ke dalam Set.
  3. DeleteSet(Set \*S, infotype Elmt) yang berfungsi untuk menghapus elemen pada Set. Karena elemen di dalam Set tidak berurut, maka elemen yang ingin dihapus dapat ditimpa dengan elemen berikutnya.
  4. Beberapa operasi yang bisa dilakukan untuk 2 Set, antara lain : SetUnion (mengembalikan Set gabungan), SetIntersection (mengembalikan Set dengan elemen yang sama), SetSymmDiff (mengembalikan Set dengan elemen yang tidak ada pada keduanya), dan SetSubtract (mengembalikan Set hasil pengurangan Set 1 dengan Set 2)
- Persoalan yang diselesaikan menggunakan ADT ini adalah persoalan untuk menyimpan nama di setiap game sehingga memastikan tidak ada nama yang duplikat. Gunakan set yang berbeda di setiap game untuk memudahkan pencatatan.
- Alasan mengapa ADT ini digunakan adalah karena elemen yang tersimpan pada Set bersifat unik sehingga hal tersebut dapat mempermudah program untuk menyimpan daftar nama pemain yang memainkan suatu game.

### 3.3 ADT map

Struktur data lain yang digunakan selanjutnya adalah struktur data Map. Map merupakan kumpulan pasangan  $\langle \text{key}, \text{value} \rangle$ , dengan nilai key yang bersifat unik. Konsepnya hampir sama seperti associative array, dimana value-nya tidak disimpan secara langsung, melainkan dengan memberikan key tertentu sesuai dengan pembagiannya. Struktur data Map diimplementasikan dalam bentuk ADT map dengan sketsa struktur data yang hampir sama seperti ADT set, yaitu :

- Memiliki tipe data Map yang terdiri dari memori tempat penyimpanan elemen bertipe `infotypeM` sebanyak 100 elemen, serta sebuah integer `Count` yang menyatakan jumlah dari elemen yang terdapat pada Map. Tipe data `infotypeM` terdiri dari sebuah key dan value bertipe integer sebagai pasangan nilai dari satu elemen Map.
- Memiliki beberapa fungsi/prosedur yang dapat digunakan untuk melakukan operasi terhadap elemen di dalam Map, seperti `CreateEmptyMap`, `IsEmptyMap`, `IsFullMap`, dan `IsMemberMap` yang sama seperti ADT stack dan ADT set. Namun, terdapat beberapa perbedaan pada operasi lain :
  1. `InsertMapSorted(Map *M, keytypeM k, valuetype v)` yang berfungsi untuk menambahkan elemen dengan key `k` dan value `v` ke dalam Map. Apabila key `k` sudah ada pada Map, maka operasi tidak dilakukan.
  2. `DeleteMap(Map *M, keytypeM k)` yang berfungsi untuk menghapus elemen dengan key bernilai `k` pada Map. Sama seperti Set, karena elemen di dalam Map tidak berurut, maka elemen yang ingin dihapus dapat ditimpa dengan elemen berikutnya.
  3. `Value(Map M, keytypeM k)` yang berfungsi untuk mengembalikan value dari `infotypeM` dengan key bernilai `k`.
- Persoalan yang diselesaikan menggunakan ADT ini adalah persoalan penyimpanan skor untuk setiap pemain yang berbeda. Di setiap game, akan digunakan Map yang berbeda untuk memudahkan pencatatan.
- Alasan mengapa ADT ini digunakan adalah karena elemen yang disimpan oleh map terdiri dari sepasang key dan value yang dapat mempermudah program untuk menyimpan daftar skor untuk setiap permainan yang ada pada BNMO.

### 3.4 ADT listlinier

Struktur data yang selanjutnya digunakan adalah struktur data Linked List. Linked List merupakan serangkaian struktur berkait yang terdiri atas sebuah node yang terkait dengan node lainnya. Node sendiri merupakan sebuah tuple yang terdiri atas sebuah nilai dengan tipe tertentu (`info`) dan sebuah penunjuk ke node lain (`next`) yang bisa saja tidak menunjuk ke node manapun sehingga bernilai `Nil`. Struktur data Linked List diimplementasikan dalam bentuk ADT listlinier dengan sketsa struktur data yang hampir sama seperti ADT-ADT sebelumnya, yaitu :

- Memiliki tipe data List yang dikenali melalui Address dari elemen pertamanya (`First`). Address sebuah penunjuk ke satu node tertentu bertipe `ElmtList` yang terdiri dari sebuah `infotype` (`info`) dan address (`next`) sebagai penunjuk ke node berikutnya.
- Memiliki beberapa fungsi/prosedur yang dapat digunakan untuk melakukan operasi terhadap elemen di dalam Linked List, seperti `CreateEmpty`, `IsEmptyList`, dan `NumberList` (untuk

menghitung jumlah elemen) yang sama seperti ADT-ADT sebelumnya. Namun, terdapat beberapa perbedaan pada prosedur/fungsi untuk operasi lain, seperti :

1. Alokasi(infotype X) dan Dealokasi(address \*P) yang memiliki fungsi untuk mengirimkan address dari alokasi sebuah infotype dan melakukan pengembalian address P.
  2. Search(List L, infotype X) yang akan mengembalikan address dari sebuah infotype dengan Info(P) = X pada Linked List. Jika tidak ada, maka fungsi akan mengirimkan Nil.
  3. InsV (Insert Value) dan DelV (Delete Value) yang berfungsi untuk menambahkan/menghapus elemen Linked List berdasarkan input infotypenya.
  4. Insert dan Delete/Del yang berfungsi untuk menambahkan/menghapus elemen Linked List berdasarkan address node-nya. Perbedaan dengan InsV/DelV adalah, pada fungsi Insert/Delete, elemen sudah dialokasi terlebih dahulu sehingga tidak perlu melakukan alokasi/dealokasi di dalam prosedur.
  5. PrintInfo(List L) yang berfungsi untuk mencetak Linked List ke layar.
  6. InversList(List \*L) yang berfungsi untuk membalik urutan elemen Linked List tanpa melakukan Alokasi/Dealokasi elemen.
  7. Konkatl(List \*L1, List \*L2, List \*L3) yang berfungsi untuk menggabungkan L1 dan L2, kemudian hasilnya disimpan pada L3 sehingga Initial State L1 dan L2 masih sama.
- Terdapat tipe data tambahan di luar tipe data List dan primitifnya, yaitu tipe data GameScoreList, terdiri dari sebuah List, yang digunakan sebagai memori tempat penyimpanan elemen, dan sebuah integer Neff yang menyatakan indeks efektif dari GameScoreList. Untuk tipe data ini, hanya terdapat 1 primitif tambahan, yaitu CreateGameScore(GameScoreList \*GL) yang berfungsi untuk membuat List Score kosong (Neff bernilai 0).
  - Persoalan yang diselesaikan menggunakan ADT ini adalah persoalan mengenai daftar game beserta skor untuk setiap pemainnya. Karena ..
  - Alasan mengapa ADT ini digunakan adalah karena memungkinkan untuk melakukan penyimpanan elemen-elemen yang ada pada Linked List tanpa harus kontigu. Dengan konsep address yang dimiliki (sebagai penunjuk ke sebuah node), kini penyisipan elemen dapat dengan mudah dilakukan terutama untuk melakukan update terhadap score di suatu game.

### 3.5 ADT listsom

Struktur data lain yang digunakan adalah struktur data List Snake on Meteor. List SoM memiliki struktur data seperti list berkait dengan variasi pencatatan elemen terakhir pada List dan adanya ADT point didalamnya. Struktur data berupa sebuah node yang terdiri atas info node yang berupa ADT point dan pointer yang menunjukkan elemen berikutnya, dan pointer pada node terakhir bernilai NULL.

Struktur data List SoM diimplementasikan dalam bentuk ADT listsom dengan sketsa struktur data yang hampir sama seperti ADT list berkait sebelumnya yaitu :

- Memiliki tipe data ListSnake yang berisi somaddress (address) dari elemen pertama (Head) dan elemen terakhir list (Tail). Tipe somaddress merupakan sebuah pointer ke



elemen list berikutnya (ElmtSnake) yang berisi sebuah infoPos yang betipe Point yang mana menyimpan integer x dan y. Lalu juga ada somaddress nextPos yang menunjukan address elemen berikutnya.

- Memiliki beberapa prosedur/fungsi yang dapat digunakan untuk melakukan operasi terhadap elemen di dalam List Snake on Meteor seperti MakePoint, IsEQPoint, IsEmptySnake, CreateEmptySnake, AlokasiSnake, DealokasiSnake, LengthSnake, SearchSnake, SearchIdxSnake, InsV, DelV, DelP, SearchObstacle. Pada dasarnya kegunaan fungsi/prosedur pada ADT ini sama dengan operasi primif pada List Linier yang sudah dijelaskan dengan tambahan operasi yakni :
  1. MakePoint(int x, int y), membentuk sebuah Point dari komponen-komponennya.
  2. IsEQPoint(Point P1, Point P2), mengirimkan true jika  $P1 = P2$  : absis dan ordinatnya sama.
  3. SearchSnake(ListSnake L, Point X), mengirimkan address dari list yang memiliki point X, NilSOM jika tidak ada
  4. SearchIdxSnake(ListSnake L, Point X), mengirimkan index dari list yang memiliki point X, NilIdx jika tidak ada.
  5. SearchObstacle(ListObstacle L, Point P), Mengirimkan true jika P ada di salah satu obstacle
- Persoalan yang diselesaikan dengan ADT ini adalah dapat mengetahui lokasi Tail Snake secara langsung serta dapat menyimpan info bertipe point lokasi.
- Alasan digunakan ADT ini adalah untuk menyimpan infotype dari sebuah node yang berupa point dan dapat mengakses elemen terakhir dari sebuah List.

### 3.6 ADT listrek

Struktur data lain yang digunakan adalah struktur data List Rekursif. List Rekursif memiliki struktur data yang sama dengan List Linier, yakni merupakan sebuah node yang terdiri atas nilai dengan tipe tertentu yang berisikan informasi (info) dan sebuah pointer yang menunjuk elemen node lain (next). List rekursif juga diakhiri oleh node yang elemen next-nya bernilai NIL. Struktur data List Rekursif diimplementasikan dalam bentuk ADT listrek dengan sketsa struktur data yang hampir sama seperti ADT list linear sebelumnya, yaitu:

- Memiliki tipe data ListRek yang dikenali melalui addrLr (address) dari elemen pertamanya (First). Tipe addrLr merupakan sebuah penunjuk ke elemen list linear (ElmtListLr) yang terdiri atas sebuah infotype (info) dan addrLr (next) sebagai penunjuk ke elemen list berikutnya.
- Memiliki beberapa prosedur/fungsi yang dapat digunakan untuk melakukan operasi terhadap elemen di dalam List Rekursif, seperti AlokasiLr, DealokasiLr, IsEmptyLr, NbElmtList, dan SearchLr yang serupa seperti ADT List Linear namun dengan implementasi algoritma yang dilakukan secara rekursif. Namun, ada beberapa primitif-primitif lain yang unik pada ADT ini, seperti:
  1. IsOneElmtLr(ListRek L) yang berfungsi untuk mengembalikan nilai boolean true apabila list hanya memiliki satu elemen.
  2. FirstElmtLr(ListRek L) yang berfungsi untuk mengembalikan nilai infotype pada elemen pertama list.

3. TailLr(ListRek L) yang berfungsi untuk membelah list dan mengembalikan list tanpa elemen pertamanya.
  4. Konso(infotypeLr e, ListRek L) yang berfungsi untuk menambahkan elemen e ke dalam list dan mengembalikan list dengan e sebagai elemen pertamanya.
  5. KonsB(ListRek L, infotypeLr e) yang berfungsi untuk menambahkan elemen e ke dalam list dan mengembalikan list dengan e sebagai elemen terakhir.
  6. ListRek Copy(ListRek L) yang berfungsi untuk menyalin list sebagai list baru (dengan alokasi memori baru) dan mengembalikan hasil penyalinan list.
  7. MCopy(ListRek Lin, ListRek \*Lout) yang berfungsi untuk menyalin list ke dalam list lain yang sudah didefinisikan sebelumnya.
  8. ListRek Concat(ListRek L1, ListRek L2) yang berfungsi untuk menggabungkan dua buah list dan mengembalikan hasil penggabungan list.
  9. MConcat(ListRek L1, ListRek L2, ListRek \*LHsl) yang berfungsi untuk menggabungkan dua buah list dan menaruhnya ke dalam list lain yang sudah didefinisikan sebelumnya.
- Persoalan yang diselesaikan dengan menggunakan ADT ini adalah tipe data Binary Tree yang memerlukan list dengan pengaksesan secara rekursif.
  - Alasan mengapa ADT ini digunakan adalah diperlukan pengaksesan sebuah struktur data yang bisa dilakukan secara rekursif.

### 3.7 ADT binTree

Struktur data lain yang digunakan adalah struktur data Binary Tree. Struktur data ini merupakan serangkaian struktur berkait yang terdiri atas sebuah node yang terkait dengan node-node lainnya. Setiap nodenya merupakan sebuah tuple yang terdiri atas sebuah nilai dengan tipe tertentu (info), sebuah penunjuk (address) ke node lain (left), dan juga penunjuk ke node lainnya (right). Elemen penunjuk ke node lainnya bisa saja kosong sehingga bernilai Nil. Struktur data ini memanfaatkan ADT List Rekursif untuk pemrosesan datanya yang dilakukan secara rekursif. Struktur data Binary Tree diimplementasikan dalam bentuk ADT bintree dengan sketsa struktur data yang hampir sama seperti ADT-ADT sebelumnya, yaitu:

- Memiliki tipe data BinTree yang dikenali melalui address dari elemen pertamanya (akar). Akar tersebut merupakan sebuah node yang memiliki informasi nilai dengan tipe data tertentu (info) dan dua buah penunjuk ke node berbeda yang bisa saja kosong.
- Memiliki beberapa fungsi/prosedur yang dapat digunakan untuk melakukan operasi terhadap elemen di dalam Binary Tree, seperti Alokasi Node, Dealokasi Node yang sama seperti ADT-ADT sebelumnya. Namun, terdapat beberapa perbedaan pada prosedur/fungsi lain, seperti:
  1. Tree(infotypeLr Akar, BinTree L, BinTree R) dan MakeTree(infotypeLr Akar, BinTree L, BinTree R, BinTree \*P) yang digunakan untuk membentuk sebuah pohon. BuildBalanceTree(int n) yang digunakan untuk membuat pohon seimbang dengan info elemen bilangan masukkan pengguna.
  2. IsTreeEmpty(BinTree P) dan IsTreeOneElmt(BinTree P) untuk mengecek apakah pohon kosong atau terdiri dari satu elemen. IsUnerLeft(BinTree P) dan IsUnerRight(BinTree P),

IsBiner(BinTree P) yang digunakan untuk mengecek apakah pohon memiliki anak kiri, kanan, atau keduanya. IsSkewRight(BinTree P) dan IsSkewLeft(BinTree P) yang digunakan untuk mengecek apakah pohon condong ke kiri atau kanan.

3. PrintPreorder(BinTree P), PrintInorder(BinTree P), PrintPostorder(BinTree P), dan PrintTree(BinTree P, int h) yang digunakan untuk menampilkan pohon ke layar dengan sesuai dengan format yang diinginkan.
  4. SearchTree(BinTree P, infotypeLr X), BSearch(BinTree P, infotypeLr X), dan InsSearch(BinTree \*P, infotypeLr X) yang digunakan untuk mencari sebuah elemen dengan info X melalui metode pencarian tertentu.
  5. NbElmtLr(BinTree P), NbDaun(BinTree P), Tinggi(BinTree P) yang digunakan untuk mencari jumlah elemen, jumlah daun, atau ketinggian dari pohon. Level(BinTree P, infotypeLr X) yang digunakan untuk mencari kedalaman dari sebuah node.
  6. AddDaunTerkiri(BinTree \*P, infotypeLr X) dan AddDaun(BinTree \*P, infotypeLr X, infotypeLr Y, boolean Kiri) yang digunakan untuk menambahkan daun.
  7. DelDaunTerkiri(BinTree \*P, infotypeLr \*X), DelDaun(BinTree \*P, infotypeLr X), DelNode(BinTree \*P), dan DelBtree(BinTree \*P, infotypeLr X) yang digunakan untuk menghapus daun, node, atau sebuah pohon biner.
  8. MakeListPreorder(BinTree P) dan MakeListLevel(BinTree P, int N) yang digunakan untuk membuat sebuah list dengan memanfaatkan pohon biner.
- Persoalan yang diselesaikan dengan menggunakan ADT ini adalah persoalan untuk membuat sebuah jalur permainan pada permainan The Glass Game.
  - Alasan mengapa ADT ini digunakan adalah karena skema pemrosesan datanya yang dilakukan secara rekursif dan bentuk datanya yang bisa bercabang.

## 4 Program Utama

Program utama BNMO diimplementasikan dalam file main.c yang mengandung seluruh file header dari ADT, Game, Driver, beserta file konfigurasi yang ada di dalamnya. Sama seperti implementasi pada Tugas Besar sebelumnya, ketika program dipanggil, akan ditampilkan splash screen BNMO yang sedang melambaikan tangan untuk menyambut pemain. Setelah tampilan screen tersebut, program akan meminta user untuk memasukkan command yang terbatas pada command START dan LOAD <filename.txt>. Jika input tidak valid, maka user akan terus diminta untuk memasukkan input sampai valid. Perbedaan START dan LOAD ada pada data file yang digunakan untuk bermain, dimana START akan menggunakan file config.txt, sedangkan LOAD <filename.txt> akan menggunakan file yang sesuai dengan input user.

Program utama BNMO diimplementasikan dalam file main.c yang mengandung seluruh file header dari ADT, Game, Driver, beserta file konfigurasi yang ada di dalamnya. Ketika program dipanggil, akan ditampilkan splash screen BNMO yang sedang melambaikan tangan untuk menyambut pemain. Setelah tampilan screen tersebut, program akan meminta user untuk memasukkan command pertama kali namun terbatas pada command START dan LOAD <filename.txt>. Jika input tidak valid, maka user akan terus diminta untuk memasukkan input sampai valid. Perbedaan START dan LOAD ada pada data file yang digunakan untuk bermain, dimana START akan menggunakan file config.txt, sedangkan LOAD <filename.txt> akan menggunakan file yang sesuai dengan input user.

Setelah berhasil masuk ke dalam Menu Utama, user akan diminta untuk memasukkan command lain yang terdefinisi di dalam BNMO. Command yang valid adalah command dengan huruf besar dan menggunakan spasi yang memisahkan antar katanya. Untuk melihat daftar perintah yang dapat dijalankan, user dapat memasukkan command “HELP” pada BNMO.

Selama user tidak memasukkan command “QUIT”, user dapat memainkan permainan di dalam BNMO dengan menggunakan perintah-perintah yang valid di dalamnya. Untuk melihat daftar permainan yang disimpan oleh BNMO, user dapat memasukkan perintah “LIST GAME”. Dari sana, user dapat memilih game yang ingin dimainkan dan memasukkan game tersebut ke dalam antrian dengan menggunakan “QUEUE GAME”. Jika user merasa bosan dengan game yang terdapat di dalam BNMO, user dapat menambahkan game sendiri dengan command “CREATE GAME”. Apabila game yang ingin ditambahkan sudah berada di dalam list game, maka program tidak akan menambahkan game tersebut dan akan meminta user untuk memasukkan command baru. User juga dapat menghapus game yang sudah ditambahkan dalam list menggunakan command “DELETE GAME”, tapi tidak dapat menghapus game yang sudah disediakan BNMO dari awal file konfigurasi dibaca.

Untuk memainkan sebuah game, user harus terlebih dahulu memastikan bahwa game yang ingin dimainkan sudah berada di dalam antrian game. Kemudian, user dapat menggunakan command “PLAY GAME” dan BNMO akan secara otomatis masuk ke dalam game yang berada di antrian pertama. Apabila user sempat salah memasukkan game ke dalam antrian atau mendadak tidak ingin memainkan game tersebut, BNMO memiliki fitur skip game yang dapat melewati antrian game sebanyak n kali (tergantung jumlah antriannya) dengan command “SKIP GAME”.

Dalam pengembangan lebih lanjut, user dapat melihat nama dan skor untuk semua game yang sudah pernah dimainkan. Papan skor akan dipisah untuk game yang berbeda dan tampilan username akan diurutkan berdasarkan skor yang paling besar. Untuk mengakses papan skor, user dapat memasukkan command “SCOREBOARD”. Karena keterbatasan memori yang dimiliki oleh BNMO, user dapat menghapus daftar skor yang ada dengan cara memasukkan command “RESET SCOREBOARD”. User pun dapat memilih papan skor game apa yang ingin dihapus.

Fitur terakhir yang disediakan oleh BNMO adalah untuk melihat daftar riwayat permainan user, baik dari data yang sudah ada dari file konfigurasi (Jika LOAD) maupun yang dimulai Start Game. Nama game paling atas yang ada pada history merupakan game paling terakhir yang baru saja dimainkan oleh user. Untuk mengakses history permainan, user dapat memasukkan command “HISTORY <n>” dengan <n> adalah jumlah permainan yang telah dimainkan yang ingin ditampilkan. Apabila n lebih besar dari jumlah permainan yang telah dimainkan, maka BNMO akan menampilkan seluruh permainan yang sudah pernah dimainkan oleh user. Sama seperti papan skor, history permainan juga dapat dihapus oleh user dengan memasukkan command “RESET HISTORY”.

Adapun daftar game yang bisa dimainkan di dalam BNMO, yaitu : Random Number Generator (RNG), Diner Dash, Hangman, Tower of Hanoi, Snake on Meteor, Air Traffic Control Game (ATC Game), dan The Glass Game. Dengan mengimplementasikan apa yang sudah dipelajari oleh sang programmer, kini BNMO akan lebih sering digunakan oleh Indra dan Doni.

## 5 Data Test

Data Test yang akan diuji kali ini berasal dari file datatest.txt yang sudah diisi oleh riwayat permainan user dan daftar skor pemain untuk setiap game. Pada file tersebut, urutan permainan yang dimainkan oleh user adalah : RNG, Snake on Meteor, Hangman.

### 5.1 HISTORY <n>

Pada data test pertama, akan diuji fitur history yang akan menampilkan riwayat permainan dari user. Ketika fitur ini dipanggil, maka program akan mengeluarkan output sebagai berikut :

```
Berikut adalah daftar Game yang telah dimainkan
1. HANGMAN
2. SNAKE ON METEOR
3. RNG

ENTER COMMAND:
```

Gambar 1. Tampilan HISTORY <n>

Jumlah n yang valid pada command ini hanyalah  $n \geq 0$ . Selain daripada itu, juga untuk tipe data yang berbeda (char atau string), maka command akan dianggap tidak valid dan user harus menginput ulang command HISTORY <n>.

### 5.2 RESET HISTORY

Data test ini akan menguji fitur untuk menghapus riwayat permainan user. Apabila history tidak jadi dihapus, maka program akan menampilkan kembali riwayat permainan yang sudah pernah dimainkan.

```
Apakah kamu yakin ingin melakukan reset history? (YA/TIDAK) : TIDAK

History tidak jadi di-reset. Berikut adalah daftar Game yang telah dimainkan :
1. RNG

ENTER COMMAND: |
```

Gambar 2. Tampilan RESET HISTORY jika tidak jadi dihapus

```
Apakah kamu yakin ingin melakukan reset history? (YA/TIDAK) : YA

Menghapus History..|
```

```
History berhasil di-reset.
ENTER COMMAND: |
```

Gambar 3. Tampilan RESET HISTORY ketika berhasil dihapus

```

Belum ada Game yang pernah dimainkan.
ENTER COMMAND:

```

Gambar 4. Tampilan HISTORY setelah riwayat berhasil dihapus

Kini, riwayat permainan sudah tidak ada, sehingga jika user melakukan RESET HISTORY, maka program tidak akan menghapus riwayat karena tidak ada yang bisa dihapus.

```

Apakah kamu yakin ingin melakukan reset history? (YA/TIDAK) : YA
Belum ada game yang pernah dimainkan. History gagal dihapus.
ENTER COMMAND: |

```

Gambar 5. Tampilan RESET HISTORY ketika history kosong

### 5.3 SCOREBOARD

Pada data test ini, akan diuji fitur papan skor untuk melihat daftar skor setiap pemain untuk setiap game. Ketika fitur ini dipanggil, maka program akan mengeluarkan output sebagai berikut

```

**** SCOREBOARD GAME RNG ****
| NAMA      | SKOR      |
|-----|
| Aku      | 7000      |

**** SCOREBOARD GAME Diner DASH ****
| NAMA      | SKOR      |
|-----|
|----- SCOREBOARD KOSONG -----|

**** SCOREBOARD GAME HANGMAN ****
| NAMA      | SKOR      |
|-----|
| Aku      | 6          |

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

**** SCOREBOARD GAME SNAKE ON METEOR ****
| NAMA      | SKOR      |
|-----|
| Kamu     | 10         |

**** SCOREBOARD GAME ATC GAME ****
| NAMA      | SKOR      |
|-----|
|----- SCOREBOARD KOSONG -----|

**** SCOREBOARD GAME THE GLASS GAME ****
| NAMA      | SKOR      |
|-----|
|----- SCOREBOARD KOSONG -----|

```

Gambar 6. Tampilan SCOREBOARD ketika dipanggil

## 5.4 RESET SCOREBOARD

Data test untuk fitur terakhir yang ada pada BNMO, adalah untuk menguji fitur yang dapat menghapus scoreboard. User dapat memilih scoreboard untuk game apa yang ingin dihapus.

```
0. ALL
1. RNG
2. Diner DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. ATC GAME
7. THE GLASS GAME
SCOREBOARD YANG INGIN DIHAPUS: 3
Menghapus Scoreboard...|

Scoreboard berhasil di-reset.
ENTER COMMAND: |
```

Gambar 7. Tampilan RESET SCOREBOARD ketika dipanggil

```
**** SCOREBOARD GAME RNG ****
| NAMA      | SKOR      |
|-----|
| Aku       | 7000      |

**** SCOREBOARD GAME Diner DASH ****
| NAMA      | SKOR      |
|-----|
|          |          |
---- SCOREBOARD KOSONG ----

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

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

**** SCOREBOARD GAME SNAKE ON METEOR ****
| NAMA      | SKOR      |
|-----|
| Kamu      | 10        |

**** SCOREBOARD GAME ATC GAME ****
| NAMA      | SKOR      |
|-----|
|          |          |
---- SCOREBOARD KOSONG ----

**** SCOREBOARD GAME THE GLASS GAME ****
| NAMA      | SKOR      |
|-----|
|          |          |
---- SCOREBOARD KOSONG ----

ENTER COMMAND:
```

Gambar 8. Tampilan SCOREBOARD setelah RESET SCOREBOARD dilakukan

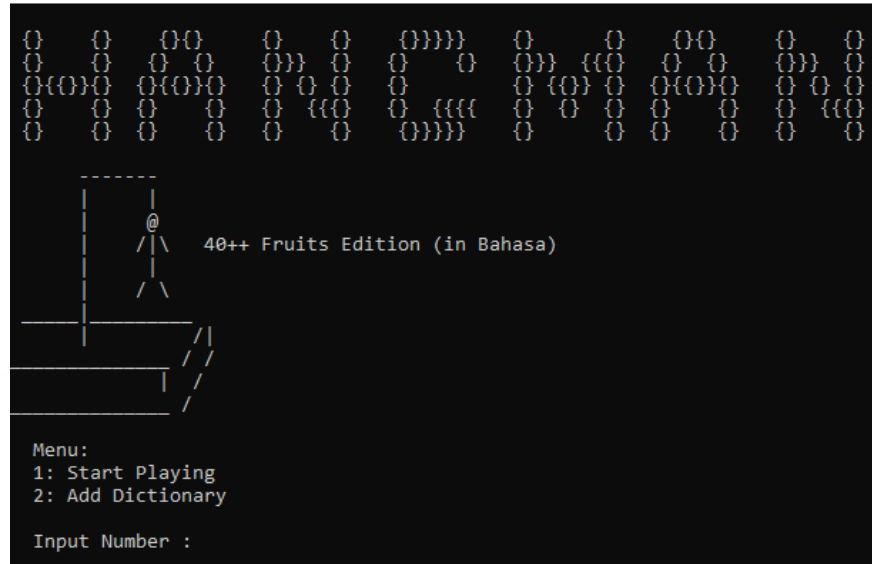
User juga bisa menghapus scoreboard untuk seluruh game yang ada pada BNMO. Namun, program akan mengirimkan konfirmasi kepada user untuk melakukan hal tersebut.

```
0. ALL
1. RNG
2. Diner DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. ATC GAME
7. THE GLASS GAME
SCOREBOARD YANG INGIN DIHAPUS: 0
APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD ALL (YA/TIDAK)? YA
Scoreboard berhasil di-reset.
```

Gambar 9. Tampilan RESET SCOREBOARD untuk menghapus seluruh scoreboard

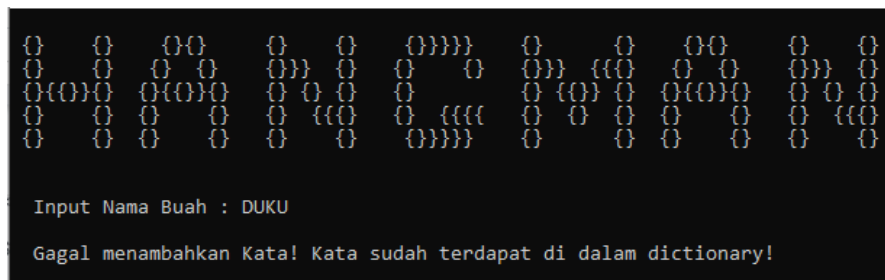
## 5.5 HANGMAN

Data test kali ini dilakukan untuk menguji game tambahan pertama yang terdapat di dalam BNMO, yaitu Hangman. Pada game ini, user diminta untuk menebak kata sudah disiapkan di dalam program (dalam kasus ini, program menyediakan daftar nama buah-buahan dalam Bahasa Indonesia).



Gambar 10. Tampilan awal game Hangman

Terdapat 2 menu yang bisa dipilih oleh user, yaitu untuk langsung mulai bermain atau menambahkan kata ke dalam daftar kata yang bisa ditebak di dalam game Hangman. User yang memilih opsi ke-2 akan diminta untuk memasukkan nama buah yang belum ada di daftar kata Hangman. Apabila sudah ada, maka program akan batal menambahkan kata dan user dikembalikan ke tampilan awal game.



Gambar 11. Tampilan menu ke-2 game Hangman

Ketika user mulai bermain, user akan dihadapkan dengan tampilan tiang Hangman seperti di bawah ini. User memiliki 10 kali kesempatan menebak kata dengan kategori buah di dalam permainan ini.





## 5.6 TOWER OF HANOI

Data test berikutnya dilakukan untuk menguji game tambahan kedua yang terdapat di dalam BNMO, yaitu Tower of Hanoi. Game ini mengharuskan user memindahkan piringan yang terdapat di tiang A ke tiang C dengan urutan yang sama. Jumlah piringan yang bisa dimainkan dibatasi sebanyak 5. Namun, user dapat mengubah jumlah piringan yang digunakan, dimana hal tersebut berpengaruh ke skor yang akan diperoleh user.

```
Jumlah piringan: 5
Masukkan jumlah piringan baru: 3
```

Gambar 15. Tampilan ToH ketika ingin mengubah jumlah piringan

```
Jumlah piringan: 3

Selamat datang di Tower of Hanoi! Gunakan perintah di bawah ini.
| CHANGE      Mengganti jumlah piringan
| START       Memulai permainan
> _

===== Petunjuk Permainan =====
1. Permainan berakhir jika semua piringan berhasil dipindah ke tiang C.
2. Hanya satu piringan yang dapat dipindahkan dalam satu putaran.
3. Setiap putaran terdiri dari mengambil piringan bagian atas dari salah satu tumpukan
   dan meletakkannya di atas tumpukan lainnya. Dengan kata lain, sebuah piringan hanya
   dapat dipindahkan jika itu adalah piringan paling atas di tumpukan.
4. Piringan yang lebih besar tidak boleh ditempatkan di atas piringan yang lebih kecil.

=====
Permainan akan dimulai dalam 3 ..._
```

Gambar 16. Tampilan awal game Tower of Hanoi

Terdapat sebuah aturan untuk game Tower of Hanoi kali ini, yaitu piringan yang lebih besar tidak dapat diletakkan di atas piringan yang lebih kecil.

```
# | |
### | |
##### | |
[===][===][===]
A B C

TIANG ASAL : B
TIANG TUJUAN : C
Tiang asal kosong!

# | |
### | |
##### | |
[===][===][===]
A B C

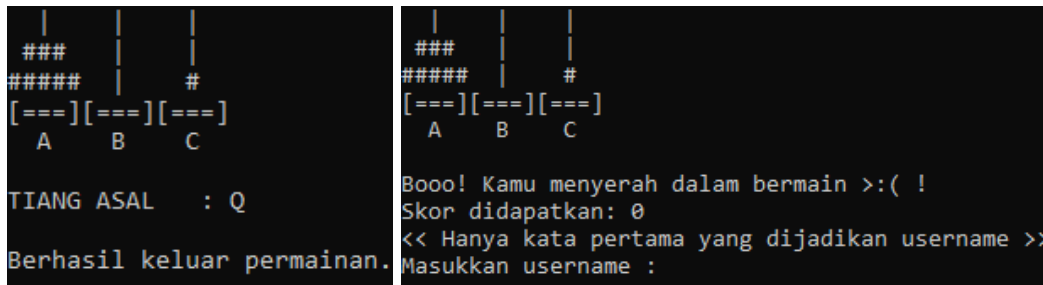
TIANG ASAL : A
TIANG TUJUAN : C
Memindahkan piringan ke C..._

| | |
### | |
##### | |
[===][===][===]
A B C

TIANG ASAL : A
TIANG TUJUAN : C
Piringan tidak dapat menimpa piringan yang lebih kecil.
```

Gambar 17. Tampilan ToH ketika user mulai bermain dan ada input yang tidak valid

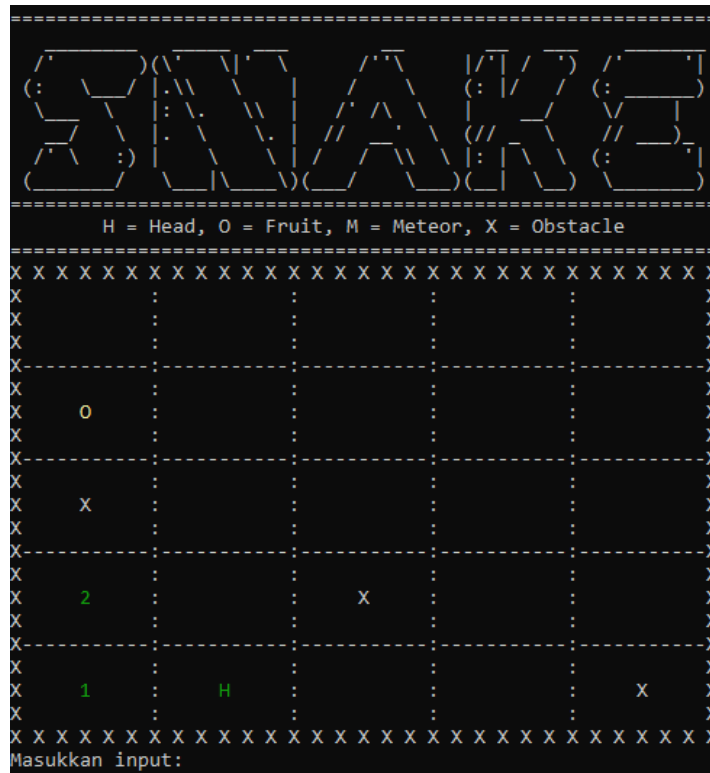
Fitur tambahan yang ada pada game ini adalah user dapat keluar secara paksa dari game. Apabila user mengaktifkan fitur quit atau berhasil menamatkan game, program akan meminta input berupa username agar skor dapat disimpan ke dalam scoreboard.



Gambar 18. Tampilan ToH ketika user keluar paksa dari game.

## 5.7 SNAKE ON METEOR

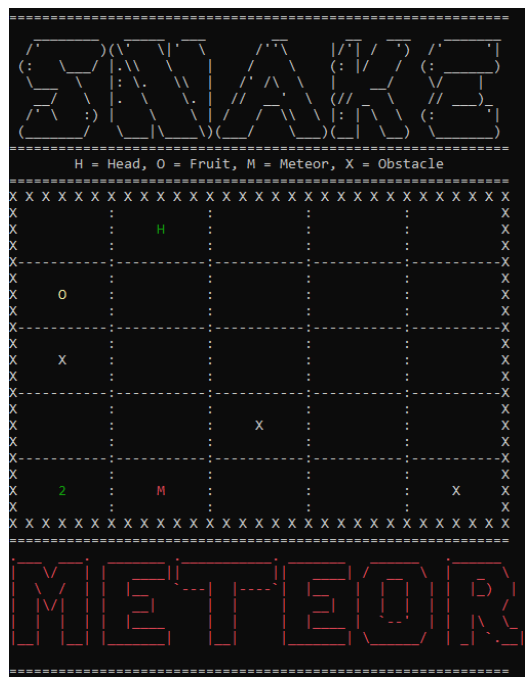
Selanjutnya, akan diuji game tambahan ketiga yang ada pada BNMO, yaitu Snake on Meteor. Game ini hampir serupa dengan Nokia Snake Game, dengan perbedaan adanya kehadiran meteor yang dapat mengenai snake tersebut. Apabila snake terkena serangan meteor tersebut adalah panjang snake akan berkurang sebanyak 1 unit.



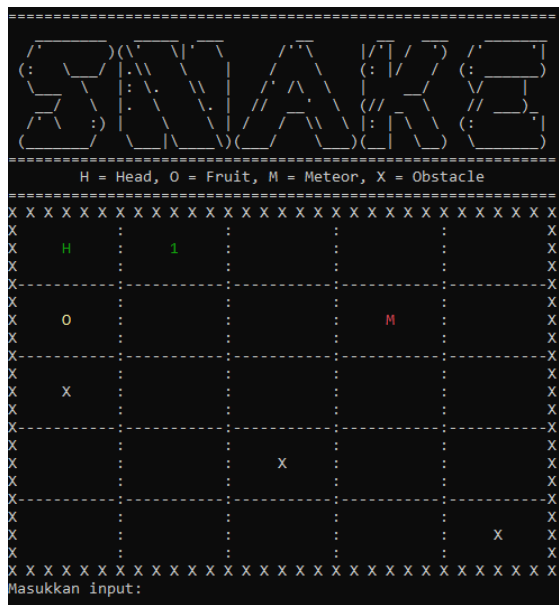
Gambar 19. Tampilan awal game Snake on Meteor

Pada tampilan tersebut, tubuh snake merupakan yang berwarna hijau dengan H sebagai penanda kepala snake. X berwarna putih merupakan obstacle yang harus dihindari, o berwarna kuning

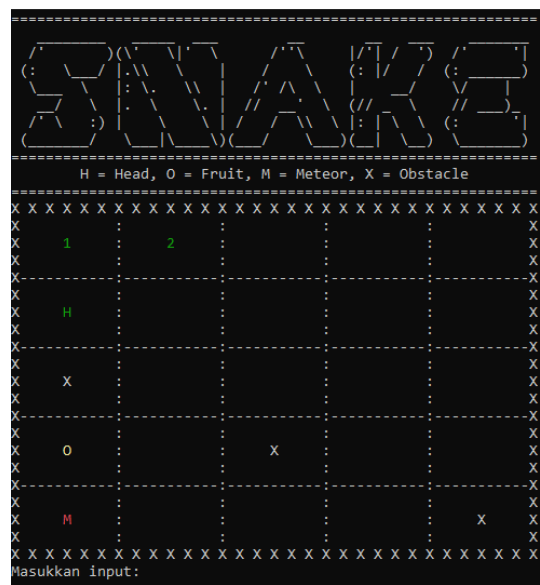
merupakan makanan snake, dan M berwarna merah merupakan meteor yang secara random dapat mengenai tubuh snake. Seperti game pada umumnya, snake dapat digerakkan dengan memasukkan huruf W/A/S/D dalam huruf besar.



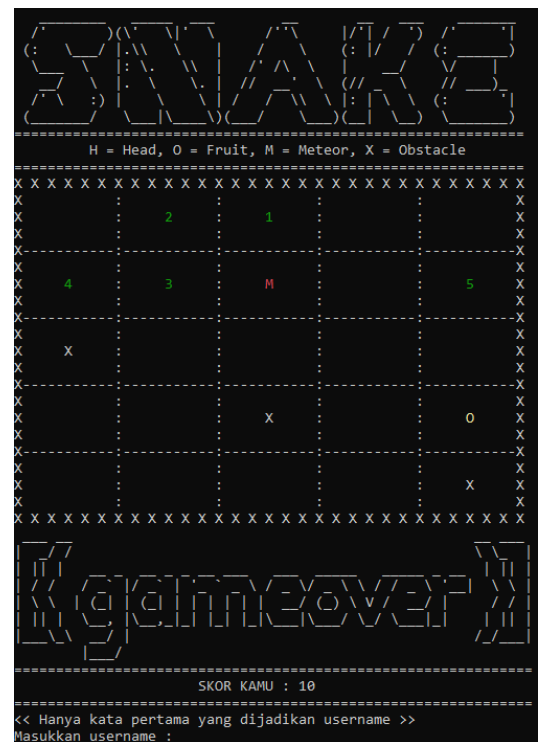
Gambar 20. Tampilan SoM ketika meteor mengenai tubuh snake



Gambar 21. Panjang snake berkurang 1 karena terkena meteor sebelumnya



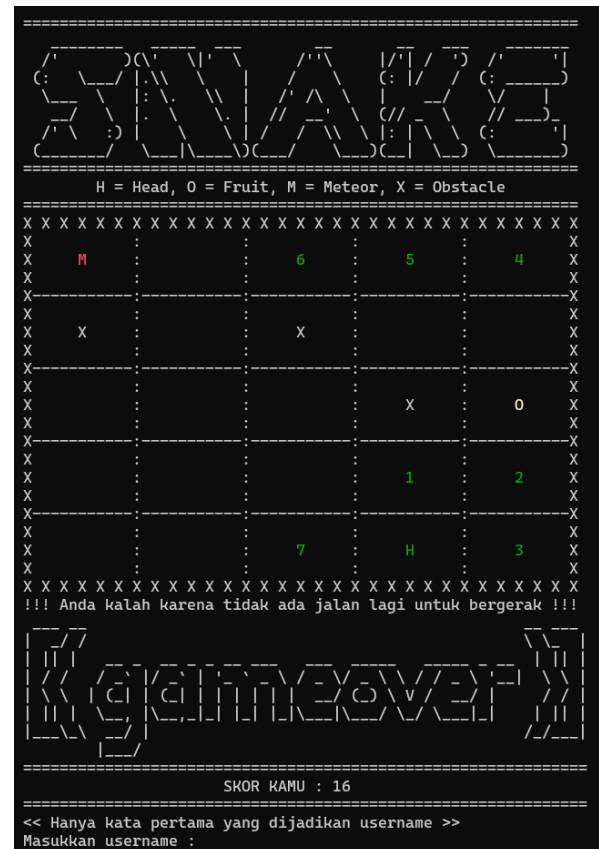
Gambar 22. Setelah mengambil makanan, panjang snake bertambah 1



Gambar 23. Tampilan SoM ketika meteor mengenai head dari snake



Gambar 24. Tampilan SoM saat snake menabrak obstacle

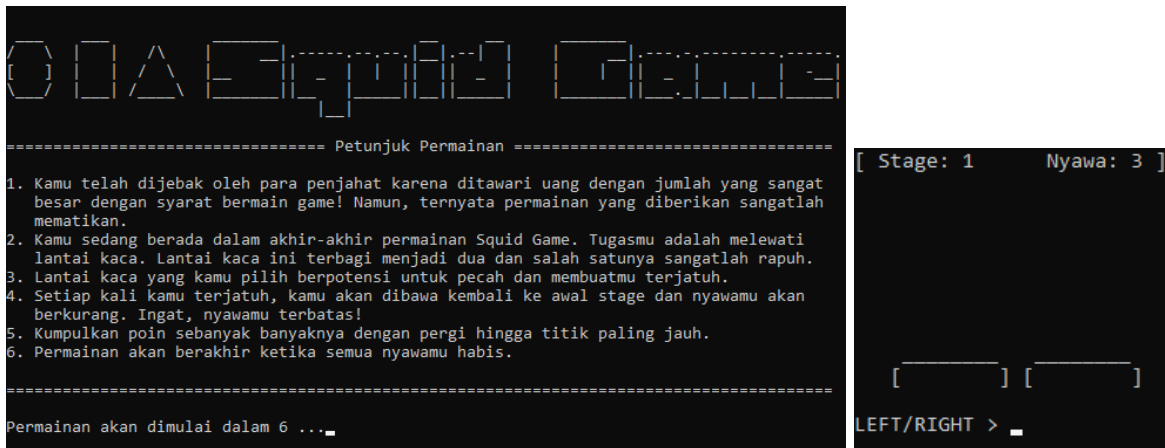


Gambar 25. Tampilan SoM saat tidak memiliki ruang untuk bergerak

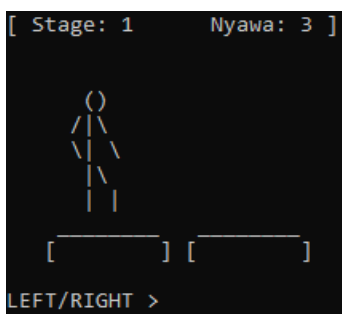
Game Snake on Meteor tidak memiliki kondisi menang secara khusus. Namun, user bisa kalah apabila bertemu dengan beberapa kondisi tertentu yakni bagian kepala terkena meteor, menabrak obstacle, dan tidak bisa lagi bergerak karena kepala dikelilingi tubuh snake. Sama seperti game-game sebelumnya, program akan meminta input berupa username agar skor dapat disimpan ke dalam scoreboard.

## 5.8 THE GLASS GAME

Data test kali ini dilakukan untuk menguji custom game yang terdapat di dalam BNMO, yaitu The Glass Game. Game ini terinspirasi dari babak terakhir K-Drama Squid Game, dimana user diberikan tantangan untuk dapat melewati lantai kaca sampai akhir jalan. User dapat memilih antara LEFT / RIGHT dan program akan secara random memiliki mana lantai kaca yang benar.



Gambar 24. Tampilan awal The Glass Game



Gambar 25. Tampilan ketika tebakan user benar

Selamat! kamu berhasil melewati babak ini.  
Skor kamu sementara adalah 3  
Siap-siap untuk menuju babak berikutnya...

Gambar 26. Tampilan ketika user berhasil lolos ke stage berikutnya

Apabila tebakan user salah, program akan membawa user ke awal stage dan user harus memasukkan jalan yang benar yang sudah berhasil ditebak sebelumnya. Permainan ini secara tidak langsung melatih daya ingat user karena seiring bertambahnya stage, maka jumlah lantai kaca yang harus ditebak juga akan semakin banyak.



Gambar 27. Tampilan ketika tebakan user salah

```
[ Stage: 4          Nyawa: 0 ]

      ()_
       /\ 
      | \ 
     . | \ 
    . /  \ .
   -/    \-
 [_____] [. ' -/]

Kamu terjatuh!
LEFT/RIGHT > RIGHT

Permainan berakhir! Skor akhir kamu: 1056

<< Hanya kata pertama yang dijadikan username >>
Masukkan username :
```

## 6 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Fitur HISTORY <n>	Memastikan bahwa program dapat menyimpan dan mencetak riwayat permainan ke layar sesuai dengan urutan permainan	Memasukkan command “HISTORY <n>” dengan n berupa integer lebih besar dari 0	HISTORY 5	Program mencetak output riwayat permainan dengan jumlah yang sesuai dengan n dimana game yang baru dimainkan terletak di paling atas	Sesuai dengan yang diharapkan
2	Fitur RESET HISTORY	Memastikan bahwa program dapat menghapus riwayat permainan yang sudah pernah dimainkan	1. Memasukkan command “RESET HISTORY” 2. Memasukkan konfirmasi apakah command ingin dieksekusi/tidak	RESET HISTORY  YA	Program mencetak output berhasil ke layar dan menghapus stack history	Sesuai dengan yang diharapkan
3	Fitur SCOREBOARD	Memastikan bahwa program dapat menyimpan dan mencetak papan skor setiap game	Memasukkan command “SCOREBOARD”	SCOREBOARD	Program mencetak output papan skor ke layar untuk setiap game yang berbeda,	Sesuai dengan yang diharapkan

					terurut berdasarkan skor yang paling tinggi	
4	Fitur RESET SCOREBOARD	Memastikan bahwa program dapat menghapus papan skor dari sebuah/seluruh game	<ol style="list-style-type: none"> <li>1. Memasukkan command "RESET SCOREBOARD"</li> <li>2. Memasukkan nomor dari scoreboard yang ingin dihapus</li> <li>3. Memberikan konfirmasi apabila user ingin menghapus seluruh scoreboard</li> </ol>	RESET SCOREBOARD  0  YA	Program mencetak output berhasil ke layar dan menghapus scoreboard dari game yang sudah dipilih	Sesuai dengan yang diharapkan
5	Hangman - Fitur Bonus	Memastikan bahwa game dapat menerima dictionary baru ke dalam list kata hangman	<ol style="list-style-type: none"> <li>1. Memasukkan angka "2"</li> <li>2. Menambahkan kata (nama buah) yang belum tersedia</li> </ol>	2  Duku	Program menuliskan Duku ke dalam file hangman.txt dan dapat dijadikan tebakan setelah dimainkan kembali	Sesuai dengan yang diharapkan
6	Hangman - Input huruf	Memastikan bahwa game dapat menerima huruf yang belum ditebak baik dalam huruf kapital/tidak	<ol style="list-style-type: none"> <li>1. Memasukkan angka "1"</li> <li>2. Menambahkan huruf sembarang</li> </ol>	1  A	Program menuliskan huruf yang dimasukkan ke dalam guessed Alphabet dan underscore akan berubah menjadi huruf A jika terdapat huruf A dalam kata yang ditebak	Sesuai dengan yang diharapkan
7	Hangman - Input tidak valid	Memastikan bahwa game hanya dapat menerima satu huruf	Memasukkan huruf lebih dari satu atau angka	a2	Program akan mencetak output untuk input satu huruf ke layar dan meminta input ulang	Sesuai dengan yang diharapkan
8	Tower of Hanoi - Fitur Bonus	Mengubah banyak piringan yang ingin dimainkan user	<ol style="list-style-type: none"> <li>1. Masukkan command "CHANGE"</li> <li>2. Masukkan jumlah piringan yang diinginkan</li> </ol>	CHANGE  3	Jumlah piringan yang akan dimainkan berubah sesuai jumlah yang dimasukkan user	Sesuai dengan yang diharapkan
9	Tower of Hanoi - Input tiang	Memindahkan piringan dari	1. Masukkan command "START"	START	Piringan paling atas pada tiang	Sesuai dengan yang diharapkan



		tiang A ke tiang C	2. Masukkan "A" untuk tiang asal dan "C" untuk tiang tujuan	A C	A berpindah ke tiang C	
10	Tower of Hanoi - Fitur Force Quit	Keluar dari permainan saat sedang memainkan permainan	1. Masukkan command "START" 2. Masukkan "A" untuk tiang asal dan "C" untuk tiang tujuan 3. Masukkan command "Q" untuk keluar dari permainan	START  A C Q	Keluar dari permainan dan user mendapatkan skor 0	Sesuai dengan yang diharapkan
11	Tower of Hanoi - Input tidak valid	Memastikan bahwa game hanya dapat menerima input yang valid	Memasukkan tiang selain A/B/C, tiang asal yang kosong, atau tiang dengan piringan yang lebih besar ke tiang tujuan dengan lebar piringan yang lebih kecil	START  B C	Program akan mencetak output mengapa command tidak valid ke layar dan minta input ulang	Sesuai dengan yang diharapkan
12	Snake on Meteor - Input gerakan	Memastikan bahwa snake akan bergerak sesuai dengan input W/A/S/D	Masukkan input "W/A/S/D"	W	Apabila tidak ada meteor atau obstacle, snake berhasil bergerak ke atas	Sesuai dengan yang diharapkan
13	Snake on Meteor - Inputan tidak valid	Memastikan bahwa game hanya dapat menerima input W/A/S/D untuk pergerakan snake	Masukkan input selain "W/A/S/D"	b	Program akan mencetak output command tidak valid ke layar dan minta input ulang	Sesuai dengna yang diharapkan
14	Snake on Meteor - Inputan valid tetapi bergerak ke arah yang tidak valid (Meteor / Badan Ular)	Memastikan bahwa snake tidak akan bergerak meskipun inputan sesuai dengan input W/A/S/D	Masukkan input "W/A/S/D" dan bermaksud bergerak ke lokasi meteor / tubuh ular	W/A/S/D	Program akan mencetak output "Anda tidak dapat bergerak ke meteor! " ATAU "Anda tidak dapat bergerak ke tubuh ular!". Lalu meminta inputan kembali	Sesuai dengna yang diharapkan
14	The Glass Game - Input tebakan	Memastikan bahwa game dapat bergerak dengan menerima input LEFT/RIGHT	Masukkan input "LEFT/RIGHT"	LEFT	Apabila lantai kaca tidak rapuh, maka user dapat berjalan ke arah kiri. Jika tidak, maka user akan	Sesuai dengan yang diharapkan

					terjatuh dan mengulang dari awal stage	
15	The Glass Game - Input tidak valid	Memastikan bahwa game hanya dapat menerima input LEFT/RIGHT	Masukkan input kata lain	left	Program akan mencetak output command tidak valid ke layar dan minta input ulang	Sesuai dengan yang diharapkan

## 7 Pembagian Kerja dalam Kelompok

NIM	Nama	Deskripsi Tugas
18221065	Josua Adriel S.	<ol style="list-style-type: none"> <li>1. Membuat prosedur dan realisasi scoreboard dan resetScoreboard</li> <li>2. Memodifikasi ADT Set dan Map untuk keperluan scoreboard</li> <li>3. Membantu membuat laporan bagian data test</li> <li>4. Melakukan debugging secara keseluruhan dan revisi kode</li> </ol>
18221121	Rozan Ghosani	<ol style="list-style-type: none"> <li>1. Membuat ADT stack, set, map, listlinier, listrek, dan binTree</li> <li>2. Membuat prosedur history dan realisasi game Tower of Hanoi</li> <li>3. Membuat custom game bonus (The Glass Game).</li> <li>4. Membuat laporan bagian Spesifikasi Tambahan untuk custom game dan lalalala.</li> <li>5. Merevisi main program</li> </ol>
18221123	Abraham Megantoro S.	<ol style="list-style-type: none"> <li>1. Membuat repository bersama.</li> <li>2. Membuat realisasi game Hangman.</li> <li>3. Membuat main program dan debugging untuk seluruh fungsi/prosedur.</li> <li>4. Membuat laporan bagian Test Script dan Fitur Tambahan.</li> </ol>
18221143	Lie, Kevin Sebastian S.T.	<ol style="list-style-type: none"> <li>1. Membuat realisasi game Snake on Meteor.</li> <li>2. Membantu pembacaan dan penulisan scoreboard dari file text kedalam Map dan sebaliknya</li> <li>3. Membuat driver ADT listsom dan laporan bagian ADT ListSoM</li> <li>4. Melakukan debugging dan revisi main program.</li> </ol>
18221157	Cathleen Laurretta	<ol style="list-style-type: none"> <li>1. Narahubung asisten dengan kelompok</li> </ol>

		<ol style="list-style-type: none"> <li>2. Membuat prosedur reset history</li> <li>3. Membuat driver ADT stack, set, map, dan list linier</li> <li>4. Membuat laporan bagian cover, Struktur Data (stack, set, map, list linier), Program Utama, Data Test, dan Test Script.</li> </ol>
--	--	--

## 8 Lampiran

### 8.1 Deskripsi Tugas Besar 2

#### Latar Belakang

**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:

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 Spesifikasi Tugas Besar 1 IF2111 2022/2023

### b. SCOREBOARD

### c. RESET SCOREBOARD

### d. HISTORY <n>

### e. RESET HISTORY

## 8.2 Notulen Rapat






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

No. Kelompok/Kelas : 06 / K01  
Nama Kelompok : -  
Anggota Kelompok (Nama/NIM) :  
1. 18221065 / Josua Adriel Sinabutar  
2. 18221121 / Rozan Ghosani  
3. 18221123 / Abraham Megantoro Samudra  
4. 18221143 / Lie, Kevin Sebastian S. T.  
5. 18221157 / Cathleen Lauretta  
  
Asisten Pembimbing : 13519064 / Aditya Bimawan

---





Asistensi III



STEI- ITB	IF2111 TB2 01 06	Halaman 28 dari 34 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.		

<b>Tanggal : 24 November 2022</b>	<b>Catatan Asistensi:</b> <ol style="list-style-type: none"> <li>1. Nama di Scoreboard ga perlu dibatasi. Print-an-nya jelek gapapa, yang penting fungsinya jalan. Nanti pas demo cukup masukkan nama yang maksimalnya 10 karakter.</li> <li>2. “Kerjainnya jangan deadliner. Lanjutin aja cara kerja yang sebelumnya karena demonya lancar. Fungsi yang lebih susah dibuat, coba dibantu aja temannya (Cth : Snake on Meteor). Yang lebih ribet itu biasanya bobotnya lebih besar, jadi jangan dibiarin temennya kerja sendiri.” - kakak asisten u.u</li> <li>3. Obstacle dari Snake on Meteor (bonus) itu ga dibatasi. Tapi kayaknya jangan satu karena kurang :D Walaupun susah dimenangkan tapi dibuat sewajarnya aja.</li> </ol>
<b>Tempat : Zoom Meeting</b>	
<b>Kehadiran Anggota Kelompok:</b> No NIM Tanda tangan  1 18221065    2 18221121    3 18221123    4 18221143    5 18221157  	
	<b>Tanda Tangan Asisten:</b>

	
--	--

#### Asistensi IV

<b>Tanggal : 01 Desember 2022</b>	<b>Catatan Asistensi:</b> <ol style="list-style-type: none"> <li>1. Pas demo ga harus bisa di Linux. Fungsinya Linux itu biar asisten bisa memeriksa hasil kerja.</li> <li>2. Biar bisa di run di Linux mungkin bisa langsung di develop dari Linux aja.</li> <li>3. Di laporan cukup tulis ADT sama Data Test yang dipake di Tubes 2 aja. Kecuali buat bagian Program Utama, command-command dari Tubes 1 ditulis ulang aja biar jelas.</li> <li>4. Spesifikasi tambahan buat bonus di game jangan lupa dimasukkan ke laporan.</li> <li>5. Drivernya juga jangan lupa biar bisa di compile di Linux karena bobotnya hampir sama kayak bisa nge compile main program.</li> </ol>
<b>Tempat : Zoom Meeting</b>	
<b>Kehadiran Anggota Kelompok:</b>	
No NIM Tanda tangan  1 18221065    2 18221121    3 18221123    4 18221143  	

<p>5 18221157</p> 	
	<p><b>Tanda Tangan Asisten:</b></p> 

### 8.3 Log Activity Anggota Kelompok

Tanggal	NIM	Nama	Aktivitas
18/11/2022	-	-	Melakukan pembagian tugas per spesifikasi fungsi/prosedur yang diperlukan
20/11/2022	18221121	Rozan Ghosani	Membuat ADT tambahan yang digunakan untuk keperluan Tugas Besar 2
20/11/2022	18221121	Rozan Ghosani	Membuat prosedur history dan memodifikasi main program serta prosedur lainnya sehingga dapat memuat list history
20/11/2022	18221123	Abraham Megantoro	Melakukan merging ke main dari pull request
21/11/2022	18221121	Rozan Ghosani	Membuat game Tower of Hanoi dan menambahkan prosedur reset history
21/11/2022	18221121	Rozan Ghosani	Melakukan revisi terhadap prosedur history (mengganti ADT List ke ADT Stack) dan menghapus prosedur reset history
21/11/2022	18221123	Abraham Megantoro	Melakukan merging ke main dari pull request
23/11/2022	18221157	Cathleen Lauretta	Membuat prosedur reset history
23/11/2022	18221123	Abraham Megantoro	Melakukan merging ke main dari pull request
24/11/2022	18221121	Rozan Ghosani	Melakukan revisi terhadap prosedur deleteGame yang terkait dengan prosedur history
24/11/2022	18221121	Rozan Ghosani	Menambahkan beberapa spek ke game Tower of Hanoi
25/11/2022	18221121	Rozan Ghosani	Menambahkan bonus game beserta ADT-nya (Tree dan ListRekursif) ke dalam program

25/11/2022	18221065	Josua Adriel	Membuat prosedur scoreboard dan resetScoreboard serta memodifikasi ADT Set & Map
25/11/2022	18221143	Lie, Kevin Sebastian	Membuat ADT List untuk game Snake of Meteor serta membuat game Snake of Meteor itu
25/11/2022	18221123	Abraham Megantoro	Melakukan merging ke main dari pull request
26/11/2022	18221123	Abraham Megantoro	Membuat game Hangman serta memodifikasi beberapa ADT dan prosedur lain pada program
26/11/2022	18221143	Lie, Kevin Sebastian	Melakukan modifikasi terhadap game-game yang sudah dibuat (return score setiap pemain)
26/11/2022	18221123	Abraham Megantoro	Melakukan merging ke main dari pull request
26/11/2022	18221157	Cathleen Lauretta	Membuat Log Activity anggota kelompok dan membuat dokumen laporan
26/11/2022	18221121	Rozan Ghosani	Menambahkan fitur quit pada game Tower of Hanoi dan melakukan revisi terhadap print scoreboard ke layar
26/11/2022	18221121	Rozan Ghosani	Melakukan revisi terhadap prosedur deleteGame (nomor game yang tidak dapat dihapus menjadi 1-7)
26/11/2022	18221143	Lie, Kevin Sebastian	Melakukan revisi terhadap game Snake on Meteor (bagian obstacle dan print ke layar)
26/11/2022	18221123	Abraham Megantoro	Melakukan merging ke main dari pull request
27/11/2022	18221143	Lie, Kevin Sebastian	Menambahkan primitif tambahan pada ADT mesin kata untuk mempermudah pengaksesan scoreboard
27/11/2022	18221143	Lie, Kevin Sebastian	Melakukan revisi terhadap prosedur scoreboard, beberapa prosedur lain dan game lain yang terkait dengan scoreboard
27/11/2022	18221123	Abraham Megantoro	Melakukan merging ke main dari pull request
27/11/2022	18221123	Abraham Megantoro	Melakukan beberapa modifikasi terhadap prosedur scoreboard, help, dan game hangman
27/11/2022	18221123	Abraham Megantoro	Menambahkan prosedur clear untuk clear screen yang bisa ditangani oleh OS Linux
27/11/2022	18221123	Abraham Megantoro	Melakukan revisi terhadap game Hangman dan prosedur resetScoreboard



27/11/2022	18221121	Rozan Ghosani	Membuat laporan bagian Spesifikasi Tambahan untuk Custom Game (The Glass Game)
27/11/2022	18221123	Abraham Megantoro	Merapikan game Hangman dan melakukan revisi terhadap prosedur clear()
27/11/2022	18221157	Cathleen Lauretta	Membuat driver untuk ADT Stack, Set, Map, dan List Linier, serta melakukan revisi terhadap prosedur resethistory
28/11/2022	18221157	Cathleen Lauretta	Membuat laporan bagian Struktur Data Stack
28/11/2022	18221065	Josua Adriel	Memperbaiki bug yang ada pada main program, game hangman, dan boolean.c
28/11/2022	18221121	Rozan Ghosani	Melakukan revisi terhadap beberapa bagian dari game Tower of Hanoi
28/11/2022	18221123	Abraham Megantoro	Melakukan merging ke main dari pull request
28/11/2022	18221123	Abraham Megantoro	Memperbaiki bug yang terdapat pada prosedur resetScoreboard
28/11/2022	18221157	Cathleen Lauretta	Membuat laporan bagian Struktur Data Set & Map
29/11/2022	18221157	Cathleen Lauretta	Membuat laporan bagian Struktur Data Linked List
29/11/2022	18221121	Rozan Ghosani	Membuat laporan bagian Struktur Data List Rekursif
01/12/2022	18221157	Cathleen Lauretta	Membuat laporan bagian Program Utama, Data Test, dan Tabel Pembagian Kerja Anggota
01/12/2022	18221123	Abraham Megantoro	Memperbaiki bug deleteGame serta merapihkan Hangman dan resetHistory
01/12/2022	18221143	Lie, Kevin Sebastian	Memperbaiki bug saat pengguna memasukkan username dan menambahkan keterangan pada Snake On Meteor
01/12/2022	18221157	Cathleen Lauretta	Merevisi Driver ADT serta memperbaiki resetHistory
01/12/2022	18221121	Rozan Ghosani	Memperbaiki bug history
01/12/2022	18221123	Abraham Megantoro	Memperbaiki mesinkata, mesinkarakter, dan program agar dapat dijalankan di Linux
01/12/2022	18221121	Rozan Ghosani	Memperbaiki bug dan delay pada permainan Tower of Hanoi, ATC, dan The Glass Game
02/12/2022	18221143	Lie, Kevin Sebastian	Memperbaiki bug RNG dan menambahkan driver listsom

02/12/2022	18221157	Cathleen Laurretta	Membuat laporan bagian Test Script
02/12/2022	18221121	Rozan Ghosani	Membuat laporan bagian Struktur Data Tree
02/12/2022	18221123	Abraham Megantoro	Membuat laporan bagian Fitur Tambahan dan Test Script, serta menghias README.md
02/12/2022	18221123	Abraham Megantoro	Melakukan revisi terhadap game Hangman (bisa handle huruf kecil), ADT mesinkata beserta drivernya
02/12/2022	18221143	Lie, Kevin Sebastian	Memperbaiki bug pada Snake on Meteor
02/12/2022	18221123	Abraham Megantoro	Melakukan merging ke main dari pull request
02/12/2022	18221143	Lie, Kevin Sebastian	Membuat laporan bagian Struktur Data List Linier untuk Snake on Meteor
02/12/2022	18221121	Rozan Ghosani	Membuat driver untuk ADT listrek dan binTree