

LAPORAN TUGAS BESAR

IF2111 Algoritma dan Struktur Data

BNMO


Dipersiapkan oleh:

Kelompok 10

Attariq Muhammad Azhar	18221043
Mochamad Syahrial Alzaidan	18221055
Iskandar Muda Rizky Parlambang	18221109
Devina Ar'raudah	18221113
Arifuddin Achmad Subagja	18221127

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		IF2111-TB2-10		<jml hlm>
		Revisi	<no revisi>	<Tgl release>

Daftar Isi

1 Ringkasan	3
2 Penjelasan Tambahan Spesifikasi Tugas	4
2.1 Game Custom ADT Tree	4
2.2 Penambahan in-game pada Hangman	4
2.3 List kata dibaca dari file pada Hangman	5
2.4 Opsi jumlah piringan pada Tower of Hanoi	5
2.5 Penambahan obstacle pada Snake on Meteor	5
2.6 Penyambungan sisi peta yang berseberangan pada Snake on Meteor	5
3 Struktur Data (ADT)	5
3.1 ADT Stack	5
3.2 ADT Set & MAP	6
3.3 ADT Linked List	6
3.4 ADT Tree	6
4 Program Utama	7
5 Algoritma-Algoritma Menarik	7
5.1 <Algoritma 1>	7
5.2 <Algoritma 2>	7
6 Data Test	7
6.1 SCOREBOARD	7
6.2 RESET SCOREBOARD	8
6.3 HISTORY <n>	10
6.4 RESET HISTORY	10
6.5 Hangman	11
6.6 Tower of Hanoi	12
6.7 Snake on Meteor	13
6.8 Game Tambahan	18
6.9 ADT Stack	18
6.10 ADT Set & Map	18
6.11 ADT Linked List	19
6.12 ADT Tree	19
7 Test Script	19
8 Pembagian Kerja dalam Kelompok	20
9 Lampiran	21

9.1 Deskripsi Tugas Besar 2	21
9.2 Notulen Rapat	22
9.3 Log Activity Anggota Kelompok	23

1 Ringkasan

Setelah BNMO milik Doni dan Indra diperbaiki, mereka mengucapkan terima kasih banyak pada 5 mahasiswa tersebut. Namun, siapa sangka ternyata Doni dan Indra belakangan ini sedang tidak memiliki banyak uang. Itulah alasan juga mengapa Doni dan Indra meminta bantuan kami, bukan servis resmi BNMO. Namun, ternyata Doni dan Indra sebenarnya sangat ingin membeli *game* baru untuk BNMO mereka. Apalagi, sebentar lagi mendekati masa UAS di ITB dan mereka sangat butuh hiburan. Oleh karena itu, kami berencana untuk memberikan *surprise* pada Indra dan Doni berupa pengembangan BNMO sekaligus membuat *game* gratis baru untuk mereka.

Dalam perencanaan yang kami buat, ada beberapa hal yang akan kami tambahkan. Kini, sebab mahasiswa ITB suka berkompetisi, kami berencana membuat fitur baru berupa Scoreboard agar mereka senang dengan BNMO. Selain scoreboard, kami menyempurnakan juga dengan menambahkan fitur Reset Scoreboard, History, dan Reset History.

Selain menambahkan fitur, kami juga menambahkan *game* baru yang lebih membuat *user* berpikir sebagaimana kesenangan anak ITB juga. Contohnya seperti Hangman, Tower of Hanoi, dan Snake on Meteor. Namun, karena kami baik hati, kami juga menambahkan suatu *game* bonus di luar perencanaan awal.

Semua fitur dan spesifikasi baru tersebut kami lampirkan penjelasannya dalam laporan ini yang juga bisa digunakan sebagai *manual book* terhadap fitur dan spesifikasi baru tersebut. Kurang lebih akan dijelaskan juga deskripsi spesifikasi baru pada tugas ini, spesifikasi tambahan (bonus), ADT yang digunakan dalam pembuatan spesifikasi, keterkaitan dalam program utama, algoritma menarik, *data test*, dan pembagian tugas dalam penyusunan spesifikasi terbaru.

Seperti sebelumnya, semua fitur dan *game* tersebut kami buat dalam basis CLI (*Command Line Interface*). Kami memanfaatkan segala ilmu yang telah dipelajari di kelas IF2111, Algoritma dan Struktur Data. Kami menambahkan beberapa ADT baru juga, seperti Linked List, Set & Map, serta Stack. Namun, karena kami suka tantangan, kami mencoba juga untuk menerapkan ADT Tree untuk membuat *game* bonus. Segala tantangan baru ini diharapkan tidak hanya menyenangkan Indra dan Doni, tetapi juga memberi pengalaman dan pengetahuan berharga bagi kami para mahasiswa STI tentunya.

Asumsi:

- Masukan *user* pada permainan *Hangman* adalah karakter, bukan *string*.
- Permainan *Hangman* akan berakhir ketika nyawa habis, bukan saat pertanyaan pertama selesai dijawab. Permainan berlanjut saat pemain telah menyelesaikan soal pertama dan nyawa belum habis.
- Pada permainan *Akinator*, permainan diasumsikan berakhir saat pemain menjawab jawaban yang salah atau jawaban yang benar hingga selesai.
- Skor bertipe *float* namun ketika dimasukkan ke dalam *save file* menggunakan *command* SAVE, semua skor, kecuali pada permainan *Tower of Hanoi*, akan bertipe *integer*.

Batasan:

- Pada permainan *Tower of Hanoi*, jumlah piringan maksimum yang bisa dimasukkan adalah 38. Apabila jumlah piringan lebih dari 38, maka tampilan akan cenderung tidak rapi.
- Jumlah jenis binatang yang ditebak di dalam permainan *Akinator* sebanyak 41 binatang dan akan diacak setiap permainan dimulai.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 Akinator (Game Custom ADT Tree)

Game Custom yang kami buat menggunakan ADT Tree adalah *game* akinator. *Game* akinator ini menggunakan ADT Tree untuk menyimpan data-data *game*. *Game* ini merupakan sebuah *game* yang menantang pemainnya untuk menjadi sebuah akinator yang menebak binatang yang diberikan oleh sistem dengan cara menjawab pertanyaan-pertanyaan dengan benar. Berikut adalah spesifikasi *game* ini:

1. Binatang diambil secara random dari kosa kata yang terdapat pada program.
2. Pemain harus menjawab pertanyaan-pertanyaan dengan benar.
3. Jika pertanyaan tersebut dijawab dengan benar, pemain lanjut ke pertanyaan selanjutnya.
4. *Game* akan berakhir apabila pemain menjawab pertanyaan dengan salah atau benar semua.

2.2 Penambahan in-game pada Hangman

Dengan keterbatasan pengembang *game*, kami membuat opsi untuk pemain juga dapat berkontribusi untuk menambahkan variasi kata sesuai tema soal yang telah kami sediakan. Dalam hal ini, pemain dapat menambahkan kata untuk ikut ditebak. Pemain bisa menambahkan kata dengan mengetik angka '2' saat pertama kali memasuki *game*. Kemudian, pemain memasukkan kata baru dan enter. Kata tersebut akan otomatis diproses untuk masuk dalam set huruf dan di lakukan *save* ke dalam *file* txt. Kata yang dimasukkan akan otomatis menjadi *uppercase* saat disimpan.

2.3 List kata dibaca dari file pada Hangman

Pada saat permainan dimulai, Hangman akan melakukan *load* variasi kata soal dari sumber kata pada *dictionary*. Variasi kata ini yang akan disimpan dalam sebuah Set untuk dipanggil secara *random* sebagai soal pada setiap kali kesempatan bermain. Hal ini dimaksudkan mengurangi panjang program dan mengefisienkan permainan saat ada kata yang ditambahkan bisa tetap tersimpan hingga permainan berikutnya.

2.4 Opsi jumlah piringan pada Tower of Hanoi

Pada saat permainan dimulai, program Tower of Hanoi akan menawarkan opsi jumlah piringan pada permainan. Angka yang dimasukkan oleh pengguna nantinya akan menjadi jumlah piringan pada permainan Tower of Hanoi. Jumlah piringan yang dapat dimasukkan bebas namun karena keterbatasan *user interface*, jumlah maksimal piringan yang optimal sebesar 38. Program akan tetap bekerja jika jumlah piringan lebih dari 38, tetapi tampilan pada layar tidak sebaik jika jumlah piringan kurang dari atau sama dengan 38.

2.5 Penambahan obstacle pada Snake on Meteor

Pada saat permainan dimulai, permainan *Snake On Meteor* akan men-generate dua *obstacle* dan akan diletakkan secara acak pada map. Apabila *snake* mengenai salah satu *obstacle*, maka permainan akan berakhir. Jumlah *obstacle* yang tidak terlalu banyak membuat permainan menjadi lebih memungkinkan untuk dimainkan lebih lama namun dengan tantangan yang lebih menarik.

2.6 Penyambungan sisi peta yang berseberangan pada Snake on Meteor

Ketika bermain *Snake On Meteor*, *snake* dapat berpindah dari satu sisi ke sisi seberangnya. Contohnya ketika kepala *snake* berada di ujung kiri sisi peta dan *player* memasukkan *command* 'a' untuk bergerak ke kiri, maka kepala *snake* akan muncul pada sisi kanan peta dengan koordinat y tetap sama. Hal yang sama akan terjadi di setiap sisi peta saat *player* memasukkan *command* yang mengarahkan *snake* ke *border* dari peta. Penyambungan sisi peta yang berseberangan ini dapat menambah durasi permainan yang akan dimainkan karena kemungkinan *snake* mati semakin lebih kecil apabila dibandingkan saat sisi peta yang berseberangan tidak disambungkan.

3 Struktur Data (ADT)

BNMO merupakan *game console* yang cukup menarik karena terdapat berbagai fitur di dalamnya. Selain fitur pengelolaan *game* yang sebelumnya telah dibuat, BNMO kini ditambahkan sedikit fitur baru untuk semakin menyenangkan penggunaanya. Tidak hanya itu, BNMO juga ditambahkan beberapa *game* baru agar semakin menarik. Namun, dalam pemrogramannya tentu membutuhkan beberapa bantuan ADT baru tentunya (*Abstract Data Type*). Dalam hal ini, kami menggunakan ADT Stack, ADT Set & Map, serta ADT Linked List. Selain itu, ADT yang sebelumnya kami implementasikan pada Tugas Besar 1, masih tetap kami gunakan juga.

3.1 ADT Stack

A. Sketsa Struktur Data

- `void CreateEmptyStack(Stack *S);`
Primitif di atas merupakan primitif yang dipakai untuk membentuk sebuah stack kosong.
- `boolean IsStackEmpty(Stack S);`
Primitif di atas merupakan fungsi yang mengembalikan true apabila stack kosong.
- `boolean IsStackFull(Stack S);`
Primitif di atas merupakan fungsi yang mengembalikan true apabila stack penuh.
- `void Push(Stack *S, infotype X);`
Primitif di atas merupakan prosedur untuk menambahkan elemen ke stack.
- `void Pop(Stack *S, infotype* X);`

Primitif di atas merupakan prosedur untuk mengeluarkan elemen teratas stack dan menyimpannya ke X.

- `Stack CopyStack(Stack stack);`

Primitif di atas merupakan fungsi untuk menghasilkan stack baru dengan elemen sama persis dengan stack pada parameter.

- `Stack ReversedStack(Stack S);`

Primitif di atas merupakan fungsi yang mengembalikan stack dengan urutan terbalik dengan S.

- `int nbElmtStack(Stack S);`

Primitif di atas merupakan fungsi yang mengembalikan jumlah elemen pada stack.

B. Persoalan yang Diselesaikan

ADT ini digunakan untuk menangani fitur History dan Reset-nya. Selain itu, ADT ini juga digunakan pada game Tower of Hanoi.

C. Alasan Pemilihan

ADT ini dipilih agar memudahkan dalam menampilkan permintaan melihat History user. Jika kita ingin mengambil n game terakhir yang dimainkan, kita cukup mengeluarkan elemen Stack ke layar secara berurutan. Aturan LIFO (Last In First Out) pada Stack cukup memudahkan kita untuk menampilkan hal ini. Untuk game Tower of Hanoi, ADT ini sangat mewakili representasi aturan game Tower of Hanoi sendiri yang hanya boleh memindahkan piringan teratas (yang paling terakhir dimasukkan) dari sebuah tiang ke tiang lainnya.

D. Implementasi

ADT Stack diimplementasikan sebagai ADT Stack dengan nama file header stack.h.

3.2 ADT Set & MAP

A. Sketsa Struktur Data

- `void CreateEmptySet(Set *S);`

Primitif di atas merupakan prosedur untuk membuat set kosong pada S.

- `boolean IsEmptySet(Set S);`

Primitif ini merupakan fungsi yang mengembalikan true apabila set S kosong.

- `boolean IsFullSet(Set S);`

Primitif ini merupakan fungsi yang mengirimkan true apabila set penuh.

- `void InsertSet(Set *S, infotypeset Elmt);`

Primitif di atas merupakan prosedur untuk menambahkan Elmt ke set S.

- `void DeleteSet(Set *S, infotypeset Elmt);`

Primitif di atas merupakan prosedur untuk menghapus Elmt dari set S.

- `boolean IsMember(Set S, infotypeset Elmt);`

Primitif di atas merupakan fungsi yang mengirimkan true apabila Elmt terdapat pada S.

- `boolean IsSubsetSet(Set SAnswer, Set SKunjaw);`

Primitif di atas merupakan fungsi yang mengecek apakah SAnswer bagian dari SKunjaw.

- `void CreateEmpty(Map *M);`

Primitif di atas merupakan prosedur untuk membuat M menjadi Map kosong.

- `void CreateEmptyLM(ListMap *LM);`

Primitif di atas merupakan prosedur untuk membuat list of Map kosong.

- `boolean IsMapEmpty(Map M);`

Primitif di atas merupakan fungsi yang mengirimkan true apabila map M kosong.

- `boolean IsLMEmpy(ListMap LM);`

Primitif di atas merupakan primitif yang mengirimkan true apabila list of map kosong.

- `boolean IsFull(Map M);`

Primitif di atas merupakan fungsi yang mengirimkan true apabila map penuh.

- `valuetype Value(Map M, keytype k);`

Primitif di atas merupakan fungsi yang mengirimkan elemen dengan key K.

- `void Insert(Map *M, keytype k, valuetype v);`

Primitif di atas merupakan prosedur untuk menambahkan elemen dengan key k dan value v ke map M.

- `void InsertMapAt(ListMap *LM, Map M, addressMap i);`

Primitif di atas merupakan prosedur untuk menambahkan map ke list of map pada index ke i.

- `void Delete(Map *M, keytype k);`

Primitif di atas merupakan prosedur untuk menghapus elemen map dengan key k.

- `void DeleteMapAt(ListMap *LM, addressMap i);`

Primitif di atas merupakan prosedur untuk menghapus elemen ke-i dari list of map.

- `boolean IsMemberMap(Map M, keytype k);`

Primitif di atas merupakan fungsi yang mengecek apakah terdapat elemen dengan key k pada map M.

- `void sortmapdesc(Map *M);`

Primitif di atas merupakan prosedur untuk mengurutkan map berdasarkan valuenya.

B. Persoalan yang Diselesaikan

ADT ini digunakan untuk menangani fitur Scoreboard dan Reset-nya. Selain itu, ADT ini sangat membantu dalam implementasi *game* Hangman.

C. Alasan Pemilihan

ADT ini dipilih karena sangat mudah manajemen skor dalam Scoreboard. Apabila nama user sudah ada, kita tinggal menambahkan skor yang baru ke skor lama pada Map dengan algoritma *search*. Namun, bila belum ada cukup tambahkan pada key baru dan value baru. Pada *game* Hangman, ADT ini berfungsi untuk menyimpan variasi

kata soal yang digunakan. Hangman cukup menggunakan fungsi random number untuk mengacak soal mana yang akan ditampilkan di setiap gilirannya.

D. Implementasi

ADT Set & Map diimplementasikan sebagai ADT Set & Map dengan nama *file* header map.h dan set.h.

3.3 ADT Linked List

A. Sketsa Struktur Data

- `boolean IsEmpty (List L);`
Primitif di atas merupakan fungsi yang mengirimkan true apabila list L kosong.
- `void CreateEmptyList (List *L);`
Primitif di atas merupakan prosedur untuk membuat list kosong.
- `address Alokasi (infotypeList X, infotypeList Y);`
Primitif di atas merupakan fungsi untuk mengalokasikan elemen.
- `void Dealokasi (address P);`
Primitif di atas merupakan prosedur untuk mendealokasi address P.
- `address Search (List L, infotypeList X, infotypeList Y);`
Primitif di atas merupakan fungsi yang mengirimkan address dengan elemen X jika ada, dan nil jika tidak ada.
- `void InsVFirst (List *L, infotypeList X, infotypeList Y);`
Primitif di atas merupakan prosedur untuk menambahkan elemen pertama dengan value X.
- `void InsVLast (List *L, infotypeList X, infotypeList Y);`
Primitif di atas merupakan prosedur untuk menambahkan elemen terakhir dengan value X.
- `void DelVFirst (List *L, infotypeList *X, infotypeList *Y);`
Primitif di atas merupakan prosedur untuk menghapus elemen pertama pada list L.
- `void DelVLast (List *L, infotypeList *X, infotypeList *Y);`
Primitif di atas merupakan prosedur untuk menghapus elemen terakhir pada list.
- `void InsertFirst (List *L, address P);`
Primitif di atas merupakan prosedur untuk menambahkan P menjadi elemen pertama list L.
- `void InsertLast (List *L, address P);`
Primitif di atas merupakan prosedur untuk menambahkan P menjadi elemen terakhir list L.
- `void InsertAfter (List *L, address P, address Prec);`
Primitif di atas merupakan prosedur untuk menambahkan P menjadi elemen setelah elemen Prec pada list L.

- `void InsertBefore (List *L, address P, address Succ);`
Primitif di atas merupakan prosedur untuk menambahkan P menjadi elemen sebelum elemen Prec pada list L.
- `void DelFirst (List *L, address *P);`
Primitif di atas merupakan prosedur untuk menghapus elemen pertama pada list L.
- `void DelLast (List *L, address *P);`
Primitif di atas merupakan prosedur untuk menghapus elemen terakhir pada list.
- `void DelP (List *L, infotypeList X, infotypeList Y);`
Primitif di atas merupakan prosedur untuk menghapus elemen X pada list L.
- `void DelAfter (List *L, address *Pdel, address Prec);`
Primitif di atas merupakan prosedur untuk menghapus elemen dengan address Pdel yang terletak setelah elemen dengan address Prec.
- `void DelBefore (List *L, address *Pdel, address Succ);`
Primitif di atas merupakan prosedur untuk menghapus elemen dengan address Pdel yang terletak sebelum elemen dengan address Succ.
- `int LengthList (List L);`
Primitif di atas merupakan fungsi yang mengirimkan panjang list L.
- `void PrintForward (List L);`
Primitif di atas merupakan prosedur untuk mencetak list L terurut dari depan..
- `void PrintBackward (List L);`
Primitif di atas merupakan prosedur untuk mencetak list L terurut dari belakang.

B. Persoalan yang Diselesaikan

ADT ini digunakan untuk menangani fitur *game* Snake on Meteor. *Game* ini pada dasarnya adalah *game* umum yang cukup lama, tetapi implementasinya akan lebih mudah dengan ADT ini.

C. Alasan Pemilihan

ADT ini dipilih karena Snake on Meteor merupakan *game* yang berbasis petak koordinat. Untuk membentuk kepala, badan, dan ekor dari sang ular, akan lebih mudah menggunakan ADT Linked List ini karena aturan permainan yang mengharuskan sang ular terus bergerak/berpindah tempat. Selain itu, jika terkena meteor (pengurangan panjang badan) atau makanan (penambahan panjang badan), akan lebih mudah mengaturnya dengan ADT ini. Cukup *insert* address yang diperlukan dan *delete* address yang tidak dibutuhkan.

D. Implementasi

ADT Linked List diimplementasikan sebagai ADT Linked List dengan nama *file* header listdp.h.

3.4 ADT Tree

A. Sketsa Struktur Data

- `address newTreeNode (ElType value);`
Primitif di atas merupakan fungsi yang mengembalikan address *tree node* yang berisi value.

- `void CreateTree (Tree* tree, address node);`
Primitif di atas merupakan prosedur untuk membuat tree kosong.
- `boolean isEmpty(BinTree tree);`
Primitif di atas merupakan fungsi yang mengembalikan true apabila tree kosong.
- `void BuildTree(Tree *t, Tree *root, int x);`
Primitif di atas merupakan prosedur untuk membangun tree dengan akar root dan elemen t dengan isi x.
- `void addLeft (BinTree p, ElType x);`
Primitif di atas merupakan prosedur untuk menambahkan elemen (anak) ke kiri dari pohon.
- `void addRight (BinTree p, ElType x);`
Primitif di atas merupakan prosedur untuk menambahkan elemen (anak) ke kanan dari pohon.

B. Persoalan yang Diselesaikan

ADT ini digunakan untuk menangani fitur *game* bonus berjudul Akinator.

C. Alasan Pemilihan

ADT ini dipilih karena game Akinator memerlukan ADT yang dapat dipakai untuk menentukan jawaban benar/salah dan digunakan berulang serta dipakai untuk menentukan hasil secara akumulasi. Fitur pada ADT Tree dapat mencukupi kebutuhan pada game Akinator sehingga dipilih untuk menjadi ADT yang dipakai pada game tersebut.

D. Implementasi

ADT Tree diimplementasikan sebagai ADT Tree dengan nama *file* header tree.h.

4 Program Utama

Program kali ini sebetulnya lebih menambahkan fitur-fitur baru pada BNMO. Selain fitur yang sudah sangat lengkap diimplementasikan pada Tugas Besar 1, kali ini kami menambahkan beberapa fitur yang fokusnya lebih untuk peningkatan kepuasan bermain *game* dengan BNMO. Fitur tersebut adalah fitur Scoreboard dan Reset Scoreboard. Fitur ini membuat pemain semakin kompetitif dalam memainkan BNMO.

Selain itu, ada juga fitur History dan Reset History yang berfungsi untuk mengecek permainan apa saja yang telah dimainkan. Ini membantu *user* untuk memilih pertimbangan *game* mana yang sekiranya jarang dimainkan dan sering. Sehingga, jika *user* ingin menghapus atau menambahkan *game* nantinya akan punya pertimbangan.

Tidak lupa juga tentunya kami buat beberapa *game* yang menarik. Ada *game* Hangman, Tower of Hanoi, Snake on Meteor, dan satu *game* bonus lainnya. Namun, kami juga tidak menghapus *game* yang telah kami buat sebelumnya, yaitu RNG dan Dinner Dash.

Seluruh *game* yang telah kami buat juga memiliki aturan main yang menarik. *Game* Hangman merupakan *game* tebak kata dengan metode huruf per huruf. User hanya diberikan maksimal sepuluh nyawa untuk melakukan tebakkan. Selain itu, Tower of Hanoi adalah *game* berbasis *stack* yang cukup menantang. *Game* ini mengharuskan pemain memindahkan tumpukan piringan ke piringan lain, tetapi dengan aturan *First In First Out*. Snake on Meteor adalah salah satu *game* lainnya. *Game* ini memerlukan pemain untuk menggerakkan ular dalam petak yang

telah disediakan. Ular tidak boleh menabrak meteor dan harus memakan makanan. Ular bisa memanjangkan tubuhnya saat mendapat makanan dan terpotong tubuhnya saat terkena meteor.

Setiap selesai bermain, disinilah Scoreboard akan di-*update*. Pemain cukup memasukkan nama pemain di setiap *game* benar-benar diakhiri. Scoreboard dan jumlah history akan otomatis tersimpan pada file Save dan bisa dipanggil jika BNMO sudah dimatikan dan dihidupkan ulang.

5 Algoritma-Algoritma Menarik

Selama pengerjaan tugas besar ini, kami membuat beberapa algoritma menarik. Algoritma tersebut dijelaskan dalam poin-poin berikut :

5.1 Algoritma Clear Screen

Algoritma ini menarik karena dapat mengidentifikasi *operating system* dari pengguna dan menyesuaikan *command* Clear Screen sesuai dengan *operating system* yang dipakai pengguna.

```
void clear(){
    #if defined(_WIN32) || defined(_WIN64)
        system("cls");
    #else
        system("clear");
    #endif
}
```

6 Data Test

Fitur-fitur yang disediakan oleh sistem akan dilakukan pengetesan dengan memberikan masukan oleh *user* dan membandingkan kesamaan keluaran dari sistem dengan yang ada pada spesifikasi tugas besar.

6.1 SCOREBOARD

Deskripsi
<i>Test</i> ini bertujuan untuk menguji input SCOREBOARD dapat menampilkan nama dan skor game, Serta menampilkan hal tersebut secara berurutan. Selain itu <i>test</i> ini digunakan juga untuk menguji apakah program mampu menambahkan skor dan nama user baru ke Scoreboard. Selain itu, juga untuk mengetahui keberjalanan fungsi ini dalam menyimpan data skor melalui tampilan Scoreboard yang diberikan pada layar.
Hasil yang seharusnya diberikan
// Misal tahap ini adalah tahap game over dari game // EIFFEL TOWER Skor akhir: 12 Nama: BNMO
ENTER COMMAND: SCOREBOARD **** SCOREBOARD GAME TOWER OF HANOI ****

NAMA	SKOR	

BNMO	12	
Finn	9	

**** SCOREBOARD GAME DINER DASH****

NAMA	SKOR	
----	SCOREBOARD KOSONG	-----

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

NAMA	SKOR	

BNMO	80	
Finn	62	

**** SCOREBOARD GAME RNG****

NAMA	SKOR	

BNMO	13	
Finn	11	

**** SCOREBOARD GAME HANGMAN****

NAMA	SKOR	

BNMO	99	
Finn	1	

Data Test

```

Kepala snake terkena meteor!

): GAME OVER :(

Skor: 10

Masukkan nama: graify

Score player graify berhasil dimasukkan!
ENTER COMMAND: █

```

Gambar 6.1 Tampilan BNMO saat user mencoba memasukkan nama yang belum dipakai.

```

Kepala snake terkena meteor!

): GAME OVER :(

Skor: 4

Masukkan nama: graify
Nama sudah ada, silakan masukkan nama yang lain.
Masukkan nama: main.character7

Score player main.character7 berhasil dimasukkan!
ENTER COMMAND: █

```

Gambar 6.2 Tampilan BNMO saat user mencoba memasukkan nama yang telah dipakai.

```

** SCOREBOARD GAME RNG **
+-----+-----+
| NAMA | SKOR |
+-----+-----+
| BNMO | 19   |
| Finn | 12   |
+-----+-----+

** SCOREBOARD GAME Diner DASH **
+-----+-----+
| NAMA | SKOR |
+-----+-----+
| Jake | 58   |
| Finn | 31   |
| Marcelline | 30 |
+-----+-----+

** SCOREBOARD GAME Tebak Kata **
+-----+-----+
| NAMA | SKOR |
+-----+-----+
| SCOREBOARD KOSONG |
+-----+-----+

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

** SCOREBOARD GAME TOWER OF HANOI **
+-----+-----+
| NAMA | SKOR |
+-----+-----+
| Marshall | 10.52 |
+-----+-----+

** SCOREBOARD GAME SNAKE ON METEOR **
+-----+-----+
| NAMA | SKOR |
+-----+-----+
| Ariq | 77   |
| graify | 10  |
| main.character7 | 4   |
+-----+-----+

** SCOREBOARD GAME Akinator **
+-----+-----+
| NAMA | SKOR |
+-----+-----+
| SCOREBOARD KOSONG |
+-----+-----+

ENTER COMMAND: █

```

Gambar 6.3 Tampilan BNMO saat user memanggil command SCOREBOARD pada program

6.2 RESET SCOREBOARD

Deskripsi

Test ini digunakan untuk memastikan input RESET SCOREBOARD dapat memproses pengaturan ulang skor. Ada beberapa percobaan terhadap variabel *game* yang dipilih (ALL/Per *game*) dan validasi yakin hapus atau tidak.

Hasil yang seharusnya diberikan

ENTER COMMAND: **RESET SCOREBOARD**

DAFTAR SCOREBOARD:

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

SCOREBOARD YANG INGIN DIHAPUS: **0**

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

Scoreboard berhasil di-reset.

ENTER COMMAND: **RESET SCOREBOARD**

DAFTAR SCOREBOARD:

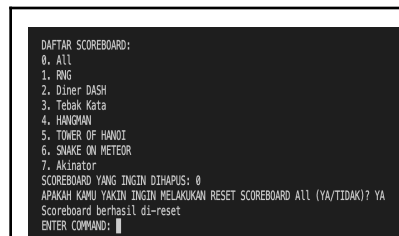
0. ALL
1. RNG
2. Diner DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR

SCOREBOARD YANG INGIN DIHAPUS: **4**

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

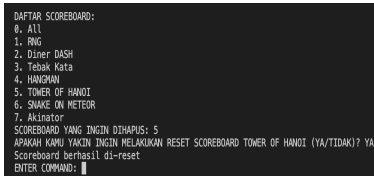
Scoreboard berhasil di-reset.

Data Test



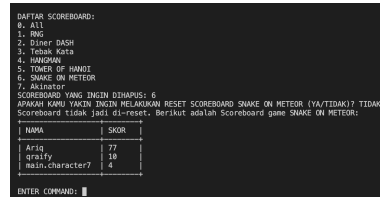
```
DAFTAR SCOREBOARD:
0. ALL
1. RNG
2. Diner DASH
3. Tebak Kata
4. HANGMAN
5. TOWER OF HANOI
6. SNAKE ON METEOR
7. Akinator
SCOREBOARD YANG INGIN DIHAPUS: 0
APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD ALL (YA/TIDAK)? YA
Scoreboard berhasil di-reset
ENTER COMMAND: █
```

*Gambar 6.4 Tampilan BNMO saat user memanggil memanggil command **RESET SCOREBOARD** dan me-reset semua scoreboard*



```
DAFTAR SCOREBOARD:
0. ALL
1. RNG
2. Diner DASH
3. Tebak Kata
4. HANGMAN
5. TOWER OF HANOI
6. SNAKE ON METEOR
7. Akinator
SCOREBOARD YANG INGIN DIHAPUS: 5
APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD TOWER OF HANOI (YA/TIDAK)? YA
Scoreboard berhasil di-reset
ENTER COMMAND: █
```

*Gambar 6.5 Tampilan BNMO saat user memanggil memanggil command **RESET SCOREBOARD** dan me-reset game tertentu*



```
DAFTAR SCOREBOARD:
0. ALL
1. RNG
2. Diner DASH
3. Tebak Kata
4. HANGMAN
5. TOWER OF HANOI
6. SNAKE ON METEOR
7. Akinator
SCOREBOARD YANG INGIN DIHAPUS: 6
APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD SNAKE ON METEOR (YA/TIDAK)? TIDAK
Scoreboard tidak jadi di-reset. Berikut adalah Scoreboard game SNAKE ON METEOR:
| NAMA | SKOR |
|-----|-----|
| Ariq | 77 |
| Greal | 50 |
| Main_Character? | 4 |
ENTER COMMAND: █
```

*Gambar 6.6 Tampilan BNMO saat user memanggil memanggil command **RESET SCOREBOARD** dan user membatalkan reset*

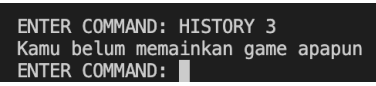
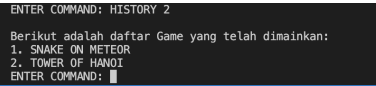
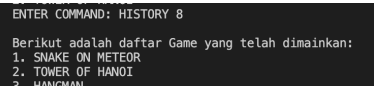
6.3 HISTORY <n>

Deskripsi

Test ini digunakan untuk menguji input HISTORY <n> mampu menampilkan daftar game apa saja yang telah user mainkan sesuai dengan variasi jumlah History yang dipanggil dengan jumlah History aktual.

Hasil yang seharusnya diberikan

ENTER COMMAND: **HISTORY 2**

Berikut adalah daftar Game yang telah dimainkan 1. EIFFEL TOWER 2. RNG		
ENTER COMMAND: HISTORY 8 Berikut adalah daftar Game yang telah dimainkan 1. EIFFEL TOWER 2. RNG 3. EIFFEL TOWER 4. RISEWOMAN 5. LUNCH SLOW		
Data Test		
 <p><i>Gambar 6.7 Tampilan BNMO saat history kosong dan user memanggil command HISTORY</i></p>	 <p><i>Gambar 6.8 Tampilan BNMO saat user memanggil command HISTORY beserta jumlahnya 2</i></p>	 <p><i>Gambar 6.9 Tampilan BNMO saat user memanggil command HISTORY beserta jumlahnya 8</i></p>

6.4 RESET HISTORY

Deskripsi
<p><i>Test</i> dilakukan untuk menguji command RESET HISTORY. Test ini akan memeriksa isi History benar-benar telah dikosongkan, setelah reset history. Command ini akan menghapus isi History seluruhnya.</p>
Hasil yang seharusnya diberikan
ENTER COMMAND: RESET HISTORY APAKAH KAMU YAKIN INGIN MELAKUKAN RESET HISTORY? YA History berhasil di-reset.
ENTER COMMAND: RESET HISTORY APAKAH KAMU YAKIN INGIN MELAKUKAN RESET HISTORY? TIDAK History tidak jadi di-reset. Berikut adalah daftar Game yang telah

dimainkan

1. EIFFEL TOWER
2. RNG
3. EIFFEL TOWER
4. RISEWOMAN
5. LUNCH SLOW

Data Test

ENTER COMMAND: RESET HISTORY

APAKAH KAMU YAKIN INGIN MELAKUKAN RESET HISTORY? (YA/TIDAK) YA
History berhasil di-reset
ENTER COMMAND: █

Gambar 6.10 Tampilan BNMO saat user memanggil command RESET HISTORY dan menginput YA

ENTER COMMAND: RESET HISTORY

APAKAH KAMU YAKIN INGIN MELAKUKAN RESET HISTORY? (YA/TIDAK) TIDAK
History tidak jadi di-reset. Berikut adalah daftar Game yang telah dimainkan
Berikut adalah daftar Game yang telah dimainkan:
1. SNAKE ON METEOR
2. TOWER OF HANOI
3. HANGMAN
ENTER COMMAND: █

Gambar 6.11 Tampilan BNMO saat user memanggil command RESET HISTORY dan menginput TIDAK

6.5 Hangman

Deskripsi

Test ini digunakan untuk menguji permainan Hangman. Permainan ini mengharuskan pemain menebak kata dengan memasukkan tiap hurufnya. Pemain hanya diberikan “nyawa” sebanyak 10 kali. Pemain akan mendapat poin sejumlah panjang kata dan akan kalah saat “nyawa” habis.

Hasil yang seharusnya diberikan

Tebakan sebelumnya: -

Kata: _ _ _ _

Kesempatan: 10

Masukkan tebakan: **A**

Tebakan sebelumnya: a

Kata: _ A _ A

Kesempatan: 10

Masukkan tebakan: **b**

Tebakan sebelumnya: ab

Kata: _ A _ A

Kesempatan: 9

Masukkan tebakan: **M**

Tebakan sebelumnya: abm

Kata: M A _ A

Kesempatan: 9

Masukkan tebakan: **F**

Tebakan sebelumnya: abmf

Kata: M A _ A

Kesempatan: 8

Masukkan tebakan: **t**

Berhasil menebak kata MATA! Kamu mendapatkan 4 poin!

Tebakan sebelumnya: -

Kata: _ _ _ _ _

Kesempatan: 8

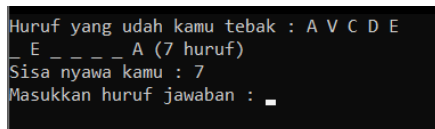
Masukkan tebakan:

Permainan berlanjut hingga kesempatan habis

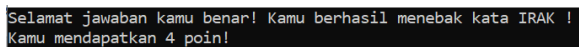
Data Test



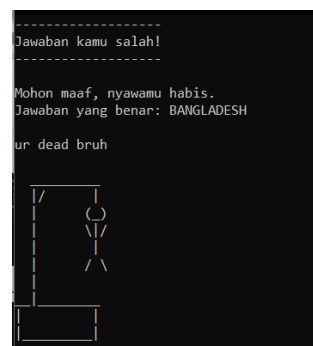
Gambar 6.12 Menu awal hangman



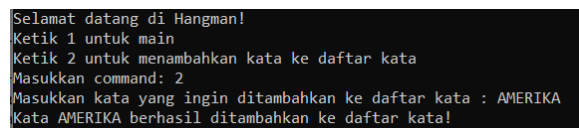
Gambar 6.13 Tampilan BNMO saat user sedang bermain Hangman



Gambar 6.14 Tampilan BNMO user berhasil menebak kata

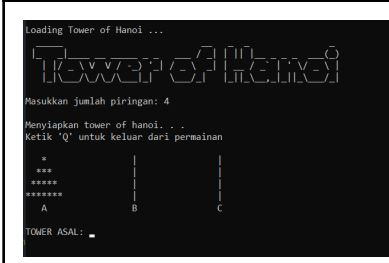
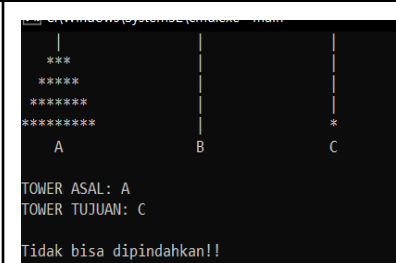



Gambar 6.15 Tampilan BNMO saat user gagal menebak kata



Gambar 6.16 Tampilan BNMO saat user menambahkan kata ke kamus

6.7 Snake on Meteor

<div>Deskripsi</div> <div> <p>Test ini digunakan untuk menguji permainan Tower of Hanoi. Permainan ini mengharuskan pemain memindahkan tumpukan lima piringan dari satu tiang ke tiang lain dengan aturan tertentu.</p> </div>								
<div>Hasil yang seharusnya diberikan</div>								
<table> <tr> <th colspan="2">Awal</th><th>Akhir</th></tr> <tr> <td colspan="2"> <pre> * *** ***** ******** ********* ***** ----- A B C </pre> <p> TIANG ASAL: A TIANG TUJUAN: B Memindahkan piringan ke B... </p> </td><td> <pre> * *** ***** ******** ********* ***** ----- A B C </pre> <p> Kamu berhasil! Skor didapatkan: 10 Nama: BNMO </p> </td></tr> </table>			Awal		Akhir	<pre> * *** ***** ******** ********* ***** ----- A B C </pre> <p> TIANG ASAL: A TIANG TUJUAN: B Memindahkan piringan ke B... </p>		<pre> * *** ***** ******** ********* ***** ----- A B C </pre> <p> Kamu berhasil! Skor didapatkan: 10 Nama: BNMO </p>
Awal		Akhir						
<pre> * *** ***** ******** ********* ***** ----- A B C </pre> <p> TIANG ASAL: A TIANG TUJUAN: B Memindahkan piringan ke B... </p>		<pre> * *** ***** ******** ********* ***** ----- A B C </pre> <p> Kamu berhasil! Skor didapatkan: 10 Nama: BNMO </p>						
<div>Data Test</div>								
 <p>Gambar 6.17 Tampilan BNMO saat user memasukkan jumlah piringan pada program</p>	 <p>Gambar 6.18 Tampilan BNMO saat user memasukkan perintah tidak valid</p>	 <p>Gambar 6.19 Tampilan BNMO saat user berhasil menyelesaikan game Tower of Hanoi</p>						

Test ini digunakan untuk memastikan permainan Snake on Meteor sudah sesuai dengan spesifikasi yang diharapkan. Permainan ini sebenarnya permainan klasik yang cukup terkenal, tetapi dibuat sedikit rumit, dengan adanya tambahan meteor yang akan mengurangi panjang snake jika mengenai tubuh snake tersebut.

Hasil yang seharusnya diberikan

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

/*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 bergerakkan 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!

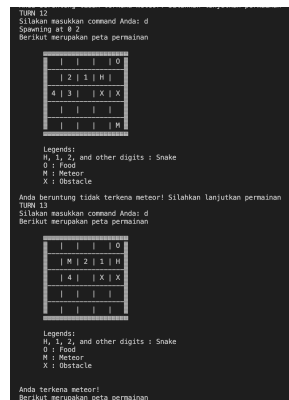
Game berakhir. Skor: 8

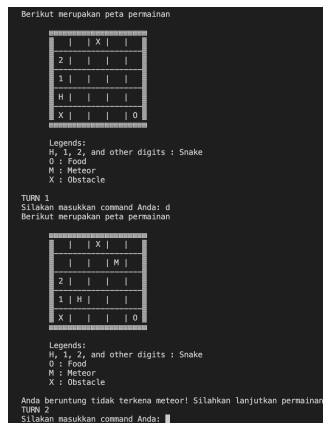
**catatan: Skor dihitung dari panjang sisa anggota tubuh yang dimiliki. Karena pada titik kepala terkena meteor, maka titik tersebut tidak akan dihitung ke dalam sisa panjang snake*

Data Test

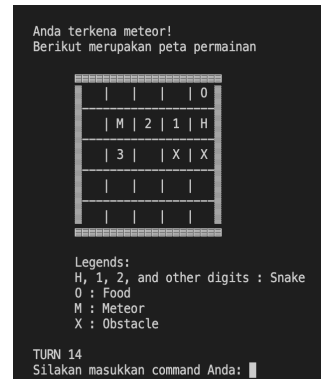


Gambar 6.20 Tampilan BNMO saat user memasukkan command yang tidak valid saat memainkan game Snake on Meteor pada program.

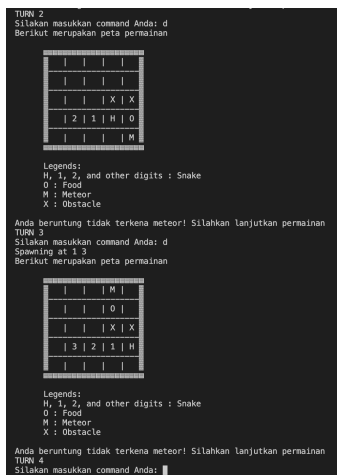




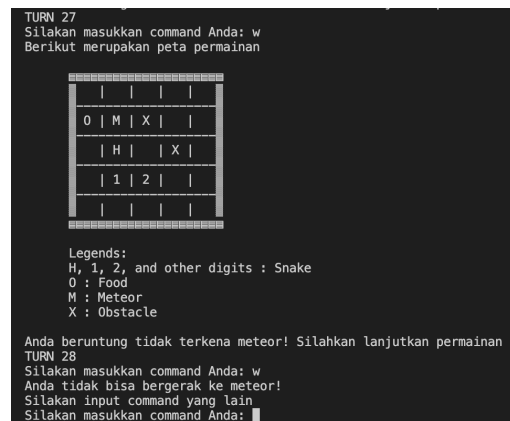
Gambar 6.21 Tampilan BNMO saat user memasukkan command yang valid saat memainkan game Snake on Meteor pada program.



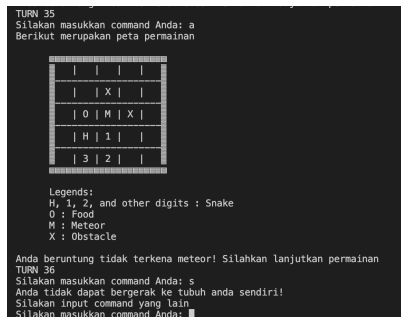
Gambar 6.23 Tampilan BNMO saat user memainkan game Snake on Meteor pada program dan badan snake terpotong oleh meteor.



Gambar 6.22 Tampilan BNMO saat user memainkan game Snake on Meteor pada program dan snake berhasil makan.



Gambar 6.24 Tampilan BNMO saat user memainkan game Snake on Meteor pada program dan menggerakkan snake ke titik koordinat meteor.



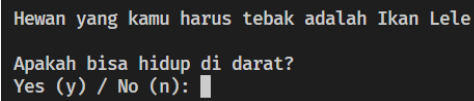
Gambar 6.25 Tampilan BNMO saat user memainkan game Snake on Meteor pada program, tapi bergerak ke tubuh sendiri



Gambar 6.26 Tampilan BNMO saat user memainkan game Snake on Meteor pada program. Berhasil makan, tapi terkena meteor pada kepala

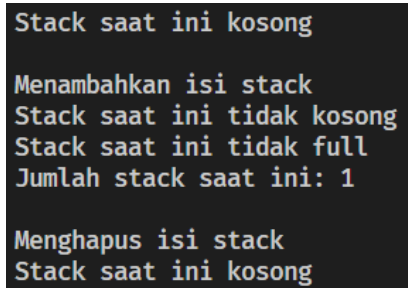
6.8 Akinator

Deskripsi
<p><i>Test</i> ini digunakan untuk memastikan permainan Akinator yang merupakan sebagai game bonus untuk ADT Tree. Permainan ini memakai ADT Tree untuk menebak benar atau salah.</p>
Hasil yang seharusnya diberikan
Loading Akinator ... Akinator /* Contoh saat bermain */ <div>Hewan yang kamu harus tebak adalah Ikan Lele Apakah bisa hidup di air? Yes (y) / No (n) :</div> <div>Hewan yang kamu harus tebak adalah Ikan Lele Apakah bisa hidup di darat? Yes (y) / No (n) :</div> <div>/* Contoh jika terdapat jawaban yang salah */ <div>+-----+ Jawaban kamu salah +-----+ Skor : 33.33</div></div> <div>/* Contoh jika yang diinput bukan y atau n */ <div>Jawaban yang diterima hanya 'y' atau 'n' Hewan yang kamu harus tebak adalah Ikan Lele Apakah bisa hidup di darat? Yes (y) / No (n) :</div></div>
Data Test

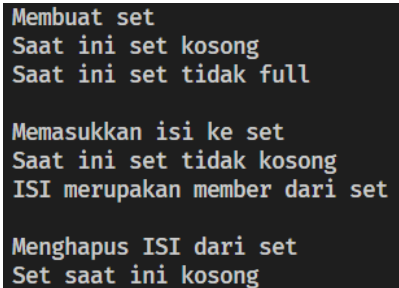
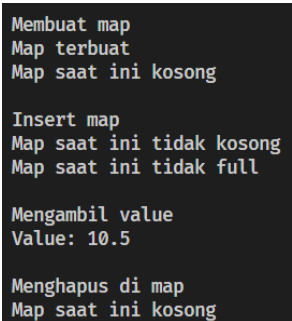


```
+-----+
|Jawaban kamu salah|
+-----+
Skor: 33.33
```

```
Jawaban yang diterima hanya 'y' atau 'n'
Hewan yang kamu harus tebak adalah Ikan Lele
Apakah bisa hidup di darat?
Yes (y) / No (n):
```

Deskripsi
<i>Test</i> ini digunakan untuk ADT Stack yang akan digunakan di dalam program.
Hasil yang seharusnya diberikan
Stack saat ini kosong Menambahkan isi stack Stack saat ini tidak kosong Stack saat ini tidak full Jumlah stack saat ini: 1 Menghapus stack Stack saat ini kosong
Data Test
 <pre> Stack saat ini kosong Menambahkan isi stack Stack saat ini tidak kosong Stack saat ini tidak full Jumlah stack saat ini: 1 Menghapus isi stack Stack saat ini kosong </pre>

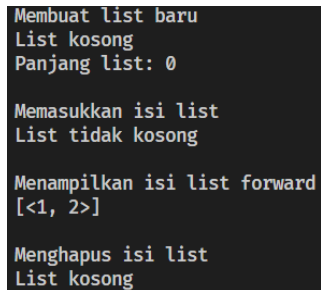
6.10 ADT Set & Map

Deskripsi	
Test ini digunakan untuk ADT Set & Map yang akan digunakan di dalam program.	
Hasil yang seharusnya diberikan	
<p>Membuat set Saat ini set kosong Saat ini set tidak full</p> <p>Memasukkan isi ke set Saat ini set tidak kosong ISI merupakan member dari set</p> <p>Menghapus ISI dari set Set saat ini kosong</p> <p>Membuat map Map terbuat Map saat ini kosong</p> <p>Insert map Map saat ini tidak kosong Map saat ini tidak full</p> <p>Mengambil value Value: 10,5</p> <p>Menghapus di map Map saat ini kosong</p>	
Data Test	
 <pre>Membuat set Saat ini set kosong Saat ini set tidak full Memasukkan isi ke set Saat ini set tidak kosong ISI merupakan member dari set Menghapus ISI dari set Set saat ini kosong</pre>	 <pre>Membuat map Map terbuat Map saat ini kosong Insert map Map saat ini tidak kosong Map saat ini tidak full Mengambil value Value: 10.5 Menghapus di map Map saat ini kosong</pre>

Gambar 6.24 Tampilan BNMO saat user mencoba menjalankan ADT Set & Map yang telah dibuat.

Gambar 6.25 Tampilan BNMO saat user mencoba menjalankan ADT Set & Map yang telah dibuat.

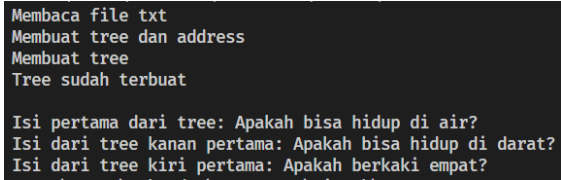
6.11 ADT Linked List

Deskripsi
<i>Test</i> ini digunakan untuk ADT Linked List yang akan digunakan di dalam program.
Hasil yang seharusnya diberikan
Membuat list baru List kosong Panjang list: 0 Memasukkan isi list List tidak kosong Menampilkan isi list forward [<1, 2>] Menghapus isi list List kosong
Data Test
 <pre>Membuat list baru List kosong Panjang list: 0 Memasukkan isi list List tidak kosong Menampilkan isi list forward [<1, 2> Menghapus isi list List kosong</pre>

Gambar 6.26 Tampilan BNMO saat *user* mencoba menjalankan ADT Linked List yang telah dibuat.

6.12 ADT Tree

Deskripsi
<i>Test</i> ini digunakan untuk ADT Tree yang akan digunakan di dalam program.
Hasil yang seharusnya diberikan
Membaca file txt Membuat tree dan address Membuat tree

<p>Tree sudah terbuat</p> <p>Isi pertama dari tree: Apakah bisa hidup di air?</p> <p>Isi dari tree kanan pertama: Apakah bisa hidup di darat?</p> <p>Isi dari tree kiri pertama: Apakah berkaki empat?</p>
Data Test
 <p>Membaca file txt Membuat tree dan address Membuat tree Tree sudah terbuat</p> <p>Isi pertama dari tree: Apakah bisa hidup di air? Isi dari tree kanan pertama: Apakah bisa hidup di darat? Isi dari tree kiri pertama: Apakah berkaki empat?</p>

Gambar 6.27 Tampilan BNMO saat user mencoba menjalankan ADT Tree yang telah dibuat.

7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Scoreboard	Memeriksa apakah BNMO mampu menampilkan nama dan skor player pada game secara berurut	Memasukkan <i>command</i> SCOREBOARD	Data Test 6.1	BNMO menampilkan <i>scoreboard</i> tiap game secara berurut.	Sesuai harapan
2	Reset Scoreboard	Menguji apakah BNMO mampu mereset skor game berdasarkan variabel game yang dipilih	Memasukkan <i>command</i> RESET SCOREBOARD, lalu memasukkan game yang ingin di reset. Selanjutnya memvalidasi, apakah yakin hapus atau tidak. Dengan menginput YA/TIDAK	Data Test 6.2	BNMO mereset <i>scoreboard</i> suatu/semua <i>game</i> sesuai masukan <i>user</i>	Sesuai harapan
3	History	Memeriksa apakah BNMO mampu menampilkan daftar game yang telah dimainkan sesuai variasi jumlah yg diinginkan	Memasukkan <i>command</i> HISTORY beserta jumlah history yang ingin ditampilkan.	Data Test 6.3	BNMO menampilkan n game terakhir yang telah dimainkan dengan n adalah angka masukan <i>user</i>	Sesuai harapan
4	Reset History	Menguji apakah BNMO mampu mereset History game yang dimainkan	Memasukkan <i>command</i> RESET HISTORY. Kemudian memvalidasi apakah yakin untuk mereset atau tidak Dengan menginput YA/TIDAK.	Data Test 6.4	BNMO mereset <i>history</i>	Sesuai harapan
5	Hangman	Menguji Permainan Hangman apakah sesuai dengan spesifikasi	Menambahkan Hangman ke dalam LIST GAME dengan menggunakan CREATE GAME. Lalu input <i>command</i> PLAY GAME ketika game	Data Test 6.5	BNMO memainkan <i>game</i> Hangman sesuai Spesifikasi.	Sesuai harapan

			Hangman telah menjadi urutan pertama dalam list game atau masukan SKIP GAME dan game selanjutnya adalah Hangman.			
6	Tower of Hanoi	Menguji Permainan Tower of Hanoi apakah sesuai dengan spesifikasi	Menambahkan Tower of Hanoi ke dalam LIST GAME dengan menggunakan CREATE GAME. Lalu input <i>command</i> PLAY GAME ketika game Tower of Hanoi telah menjadi urutan pertama dalam list game atau masukan SKIP GAME dan game selanjutnya adalah Tower of Hanoi.	Data Test 6.6	BNMO dapat memainkan <i>game</i> Tower of Hanoi sesuai Spesifikasi.	Sesuai harapan
7	Snake on Meteor	Menguji Permainan Snake on Meteor apakah sesuai dengan spesifikasi	Menambahkan Snake on Meteor ke dalam LIST GAME dengan menggunakan CREATE GAME. Lalu input <i>command</i> PLAY GAME ketika game Snake on Meteor telah menjadi urutan pertama dalam list game atau masukan SKIP GAME dan game selanjutnya adalah Snake on Meteor.	Data Test 6.7	BNMO memainkan <i>game</i> Snake On Meteor sesuai Spesifikasi.	Sesuai harapan
8	Akinator	Menguji permainan Akinator apakah dapat mengimplementasikan ADT Tree dengan baik	Menambahkan Akinator ke dalam LIST GAME dengan menggunakan CREATE GAME. Lalu input <i>command</i> PLAY GAME ketika game Akinator telah menjadi urutan pertama dalam list game atau masukan SKIP GAME dan game selanjutnya adalah Akinator.	Data Test 6.8	Permainan Akinator dapat mengimplementasikan ADT Tree dengan baik	Sesuai harapan
9	ADT Stack	Menguji ADT Stack apakah berfungsi dengan benar	Melakukan testing di driver Stack.	Data Test 6.9	Menghasilkan kalimat yang seharusnya muncul di terminal.	Sesuai harapan
10	ADT Set & Map	Menguji ADT Set & Map apakah berfungsi dengan benar	Melakukan testing di driver Set & Map.	Data Test 6.10	Menghasilkan kalimat yang seharusnya muncul di terminal.	Sesuai harapan
11	ADT Linked List	Menguji ADT Linked List apakah berfungsi dengan benar	Melakukan testing di driver Linked List.	Data Test 6.11	Menghasilkan kalimat yang seharusnya muncul di terminal.	Sesuai harapan
12	ADT Tree	Menguji ADT Tree apakah berfungsi dengan benar	Melakukan testing di driver Tree.	Data Test 6.12	Menghasilkan kalimat yang seharusnya	Sesuai harapan

					muncul di terminal.	
--	--	--	--	--	---------------------	--

8 Pembagian Kerja dalam Kelompok

No	Fitur/ADT	NIM Coder	NIM Tester
1	ADT Stack	18221113	18221043, 18221127, 18221109, 18221055
2	ADT Set & Map	18221109, 18221127	18221043, 18221113, 18221055
3	ADT Linked List	18221043	18221127, 18221109, 18221113, 18221055
4	ADT Tree	18221109	18221043, 18221127, 18221113, 18221055
5	Main Menu	18221043	18221127, 18221109, 18221113, 18221055
6	Scoreboard	18221109	18221043, 18221127, 18221113, 18221055
7	Reset Scoreboard	18221109	18221043, 18221127, 18221113, 18221055
8	History <n>	18221113	18221043, 18221127, 18221109, 18221055
9	Reset History	18221113	18221043, 18221127, 18221109, 18221055
10	Hangman	18221127	18221043, 18221109, 18221113, 18221055
11	Tower of Hanoi	18221055	18221127, 18221109, 18221113, 18221043
12	Snake on Meteor	18221043	18221127, 18221109, 18221113, 18221055
13	Akinator	18221109	18221043, 18221127, 18221113, 18221055

9 Lampiran

9.1 Deskripsi Tugas Besar 2

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

a. System Mechanics

1. About the System

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

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

2. Main Menu

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

3. Command

Terdapat beberapa aturan umum command yang digunakan:

1. Semua command yang valid harus berupa huruf kapital. Mahasiswa dibebaskan apabila ingin melakukan handling terhadap huruf kecil.
2. Command yang tidak sesuai dengan ketentuan yang disebutkan pada bagian dibawah dianggap tidak valid. Handling command tidak valid dibebaskan kepada mahasiswa (boleh meminta ulang command yang sama atau meminta command baru)

Pada setiap giliran, pemain dapat memasukkan command-command berikut:

a. Command Dasar

Semua command yang terdapat pada 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

b. Spesifikasi Game

1. RNG

Dijelaskan pada Spesifikasi Tugas Besar 1 IF2111 2022/2023.

2. Diner Dash

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

3. Hangman

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

- Pada awal permainan program menentukan sebuah kata yang harus ditebak oleh pemain. List kata dibebaskan kepada mahasiswa. Boleh ditentukan di *code*. Kemudian, pemain diberikan 10 kesempatan untuk melakukan penebakan huruf yang tidak terdapat dalam kata.
- Pada setiap giliran, pemain menebak satu huruf yang terdapat pada kata tersebut. Apabila huruf tebak terdapat dalam kata, maka huruf yang sudah tertebak akan ditampilkan pada layar. Apabila salah, maka pemain kehilangan satu kesempatan. Pemain tidak boleh menebak huruf yang sudah ditebak sebelumnya pada kata yang sama.
- Apabila pemain berhasil menebak suatu kata, maka pemain tersebut diberikan poin sesuai dengan panjang kata yang berhasil ditebak, kemudian program memberikan kata baru yang harus ditebak oleh pemain dengan jumlah kesempatan yang tersisa.
- Permainan akan berlanjut hingga pemain kehabisan kesempatan untuk menebak huruf yang salah.

4. Tower of Hanoi

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

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

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.
- 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 vertikal yang sama. Maksud dari menurun adalah secara skalar (Cth: Dari angka 3 ke angka 2, dari angka 2 ke 1, dst)
- 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:

- a. Secara umum, ekor *snake* akan bertambah pada titik ordinat yang sama dengan tail, tetapi dengan titik koordinat satu sebelum tailnya (Apabila tail berada pada titik $\langle x,y \rangle$, 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 pertambahan akan dilakukan pada titik ordinat satu sebelum ekor *snake* sekarang, tetapi pada titik koordinat yang sama (Apabila tail berada pada titik $\langle x,y \rangle$, maka pertambahan 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 pertambahan 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 pertambahan 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 pertambahan 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 pertambahan 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)
 - Meteor: Setiap putaran setelah permainan berhasil di generate (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.

- 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*.
 - 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
 - Catatan:
 1. Perhatikan dan validasi *input* gerak dari *snake*
 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*.
6. Game tambahan/ Buatlah permainan
- Game buatan pemain yang dibuat menggunakan command CREATE GAME akan langsung selesai dan masuk ke tahap game over dengan skor akhir berupa integer random.

c. Daftar ADT yang Digunakan

1. ADT Stack
ADT ini digunakan untuk merepresentasikan urutan dari permainan yang telah dimainkan, dengan TOP adalah permainan yang baru saja dimainkan dan digunakan untuk permainan Tower of Hanoi.
2. ADT Set & Map
ADT Set digunakan untuk menyimpan nama di setiap *game* sehingga memastikan tidak ada nama yang duplikat. Gunakan set yang berbeda di setiap *game* untuk memudahkan pencatatan. Sedangkan, ADT Map digunakan untuk menyimpan skor untuk setiap nama. Gunakan Map yang berbeda di setiap *game* untuk memudahkan pencatatan. Implementasi map menggunakan list dengan elemennya berupa struct yang berisi key dan value.
3. ADT Linked List
ADT ini digunakan untuk mengimplementasikan *game snake on meteor*. Tipe data yang ditampung dalam *linked list* berupa *point* yang menyimpan nilai $\langle x, y \rangle$. *Linked list* minimal digunakan pada representasi lokasi badan *snake*.

d. Bonus





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

1. Game custom menggunakan ADT Tree
Mahasiswa dibebaskan menambahkan jenis permainan dengan mengimplementasikan ADT Tree. Permainan harus memiliki output berupa score yang bertipe integer. Permainan tidak boleh menggunakan library selain dari library yang telah ditentukan.
2. Penambahan fitur pada game Hangman
 - o Penambahan in-game dictionary. Terdapat 2 menu ketika memainkan permainan Hangman, yaitu bermain langsung atau menambah kata-kata ke dalam kamus/ list kata yang berada dalam daftar kata.


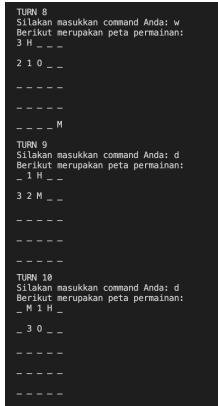
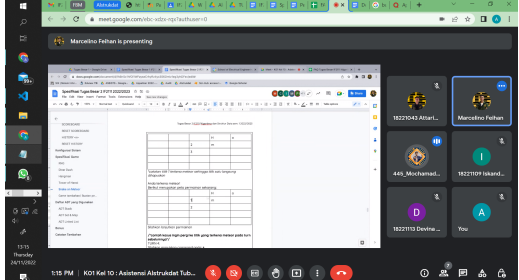

- List kata dibaca dari *file*. Di awal permainan, program membaca list kata dari file. Nama file dibebaskan oleh program (tidak di-*input* oleh pengguna). Format *file* juga dibebaskan asalkan menunjukkan daftar kata yang akan dibaca. Setelah permainan selesai/*game over*, *list* kata disimpan kembali ke *file* karena *list* kata mungkin saja bertambah.
- 3. Penambahan fitur pada game Tower of Hanoi
 - Opsi jumlah piringan. Sebelum permainan dimulai, akan diminta opsi jumlah piringan yang digunakan. Kemudian, skor yang diperoleh akan disesuaikan dengan jumlah piringan pada game (Pemain dengan jumlah piringan yang lebih sedikit akan memperoleh nilai maksimal lebih rendah daripada pemain dengan jumlah piringan yang lebih banyak).
- 4. Penambahan fitur pada game Snake on Meteor
 - Penambahan obstacle. Ketika Kepala dari *snake* mengenai obstacle, maka permainan berakhir. *Obstacle* muncul di awal permainan dan tidak dapat ditembus oleh *snake*. Selain itu, makanan juga tidak dapat muncul pada titik yang memiliki *obstacle*.
 - Menghubungkan sisi peta yang berseberangan. Ketika kepala *snake* melewati sisi atas peta, maka kepala *snake* akan muncul dari sisi bawah peta. Hal yang sama berlaku ketika kepala *snake* melewati sisi kiri/ kanan peta, maka kepala *snake* akan muncul dari sisi yang berlawanan. GIF dibawah merupakan visualisasi poin spek ini. Anggap saja ukuran map 5 x 5.

9.2 Notulen Rapat


A. Asistensi I





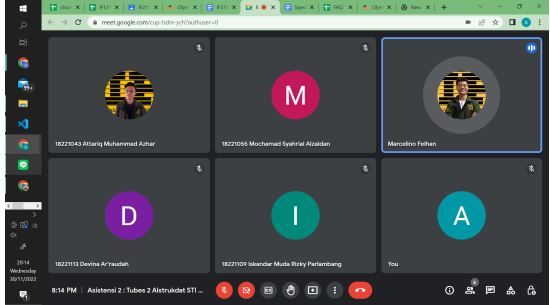

Tanggal : 24 November 2022	Catatan Asistensi:
Tempat : Google Meet	
Kehadiran Anggota Kelompok:	
1 18221055 	
2 18221109 	
3 18221113 	1. Saya masih kurang mengerti dengan maksud jawaban ka Graciella pada FAQ tentang “akan tersambung lagi setelah n step sebanyak badan setelah badan yg terkena meteor”. Apa maksud dari ‘n’ step tersebut?
4 18221043 	2. Apakah ketika men-generate badan snake di awal game badan dapat terpisah di bagian kiri dan kanan atau atas dan bawah? Kalo ga ngerjain bonus, ga boleh.
5	3. Titik spawn meteor untuk setiap turn berbeda-beda, namun apabila meteor spawn di titik makanan apakah masih valid?

1. Saya masih kurang mengerti dengan maksud jawaban ka Graciella pada FAQ tentang “akan tersambung lagi setelah n step sebanyak badan setelah badan yg terkena meteor”. Apa maksud dari ‘n’ step tersebut?
2. Apakah ketika men-generate badan snake di awal game badan dapat terpisah di bagian kiri dan kanan atau atas dan bawah?
Kalo ga ngerjain bonus, ga boleh.
3. Titik spawn meteor untuk setiap turn berbeda-beda, namun apabila meteor spawn di titik makanan apakah masih valid?

<p>18221127</p> 	 <p>4. Di Hangman diasumsikan masukan selalu satu huruf atau ngga ya kak? Asumsi aja selalu satu huruf gapapa</p> <p>5. Tampilan Tower of Hanoi itu dibebasin kan kak? Ga harus kaya di spesifikasi? Iya bebas</p>
	<p>Tanda Tangan Asisten:</p> 

B. Asistensi II

<p>Tanggal : 30 November 2022</p>	<p>Catatan Asistensi:</p>
<p>Tempat : Google Meet</p>	<p>Progress udah berapa?</p>
<p>Kehadiran Anggota Kelompok:</p> <p>1</p> <p>18221055</p>  <p>18221109</p>	<p>80-90% kak</p> <p>Laporan gimana? Udah tinggal data case sih kak</p> <p>Nanti asumsi jangan lupa dituliskan di Bagian I laporan. Asumsi itu bebas, boleh terkait data <i>type</i> atau yang lain.</p> <p>Itu ditulis dimana sih kak? Di bab I boleh kek ditulis “asumsi : “, “batasan : “, dll. gitu</p>

 3 18221113  4 18221043  5 18221127 	<p>Contoh asumsi lagi, ya gitu lah ya. Semangat!!</p>
	<p>Tanda Tangan Asisten:</p> 

C. Meeting Kelompok

No	Hari/Tanggal	Notulensi	Kehadiran Anggota Kelompok
1	Minggu, 20 Oktober 2022	Diskusi terkait spesifikasi dan ADT program, serta pembagian kerja	1. Attariq Muhammad Azhar 2. Mochamad Syahrial A
2	Minggu, 27 Oktober 2022	Diskusi terkait fungsi masing-masing dan progress report terkait fitur serta game BNMO	3. Iskandar Muda Rizky Parlambang 4. Devina Ar'raudah 5. Arifuddin Achmad Subagja

9.3 Log Activity Anggota Kelompok

No	Jadwal	Keterangan
1	Minggu, 20 November 2022	Meeting Kelompok
2	Kamis, 24 November 2022	Asistensi 1
3	Minggu, 27 November 2022	Meeting Kelompok
4	Senin, 28 November 2022	Deadline pengerjaan fungsi masing-masing
5	Selasa, 29 November 2022	Gabungin dan debugging Program
6	Rabu, 30 November 2022	Asistensi 2
7	Kamis, 1 Desember 2022	Penyusunan Laporan
8	Jumat, 2 Desember 2022	Pengumpulan Laporan
9	TBD	Demo Program