

LAPORAN TUGAS BESAR

IF2111 Algoritma dan Struktur Data


BNMO

Dipersiapkan oleh:

KELOMPOK 05

Ahmad Rizki	18221071
Nadine Aliya Putri	18221081
Clara Alrosa Fernanda Sinaga	18221099
Pramaditya Fajri Migfar	18221111
Naura Valda Prameswari	18221173

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung
Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		IF2111-TB1-05		39
		Revisi	00	2/12/2022

Daftar Isi

1 Ringkasan	4
2 Penjelasan Tambahan Spesifikasi Tugas	5
2.1 Game Treasure Hunt	5
2.2 Perhitungan Skor Game Tower of Hanoi	5
3 Struktur Data (ADT)	5
3.1 ADT stackGame	5
3.2 ADT binTree	6
3.3 ADT mapGame	6
3.4 ADT linkedlistPoint	7
3.5 ADT stackTOH	7
4 Program Utama	8
4.1 Main Menu	8
4.2 Pembacaan dan Inisialisasi File Konfigurasi	8
4.3 Pembacaan Command	8
4.4 Command Dasar	9
4.5 Command “SCOREBOARD”	9
4.6 Command “RESET SCOREBOARD”	9
4.7 Command “HISTORY <n>”	9
4.8 Command “RESETHISTORY”	9
4.9 Game Hangman	10
4.10 Game Snake on Meteor	10
4.11 Game Tower of Hanoi	11
4.12 Game ADT TREE “Treasure Hunt”	11
5 Algoritma-Algoritma Menarik	12
5.1 Build Balance Tree dengan Random Number Tiap Node	12
6 Data Test	14
6.1 Data Test 1	14
6.2 Data Test 2	15
6.3 Data Test 3	16
6.4 Data Test 4	17
6.5 Data Test 5	18

6.6 Data Test 6	18
6.7 Data Test 7	19
6.8 Data Test 8	19
6.9 Data Test 9	20
6.10 Data Test 10	20
6.11 Data Test 11	21
6.12 Data Test 12	21
6.13 Data Test 13	22
6.14 Data Test 14	22
6.15 Data Test 15	23
6.16 Data Test 16	23
6.17 Data Test 17	24
6.18 Data Test 18	24
6.19 Data Test 19	24
7 Test Script	27
8 Pembagian Kerja dalam Kelompok	28
9 Lampiran	29
9.1 Deskripsi Tugas Besar 2	29
9.2 Notulen Rapat	32
9.3 Log Activity Anggota Kelompok	36

1 Ringkasan

Indra dan Doni yang merupakan pemilik dari robot *video game console* Binomo merasa puas karena fitur-fitur dasar Binomo yang tadinya mengalami kerusakan sudah berhasil diperbaiki oleh 5 temannya yang berada di jurusan Sistem Teknologi dan Informasi. Namun, karena merasa kurang puas, Indra dan Doni meminta kembali bantuan dari 5 temannya untuk menambahkan beberapa fitur serta permainan pada BNMO agar *video game console* milik mereka tambah canggih.

BNMO merupakan sebuah *robot game console* yang dapat menjalankan berbagai jenis permainan. Binomo diprogram menggunakan bahasa C dan merupakan permainan yang berbasis CLI (*command-line interface*). Program Binomo ini menggunakan berbagai struktur data terkait list, array, mesin karakter, mesin kata, *queue*, *stack*, *set* & *map*, serta *linked list* agar dapat memenuhi kebutuhan program Binomo. Alur program Binomo dimulai dari penulisan *command* start yang akan memulai program Binomo. Setelah itu, masukan *command load* agar dapat memasukan nama file yang akan dibuka. Selain itu, kita juga dapat memainkan game, menambahkan game, menghapuskan game, serta mengurutkan game yang akan dimainkan sesuai dengan *command* yang diisi. Terdapat juga fitur tambahan berupa tampilan *history* dan *scoreboard* dari game. Kita juga dapat melakukan *reset* pada *history* dan *scoreboard* yang sudah ada. Program akan berakhir ketika *command* quit dimasukkan. Sebelum itu, kita dapat menyimpan *state game* saat ini ke dalam suatu file dengan memasukkan *command* save.

Tugas besar ini bertujuan untuk menambah wawasan mahasiswa tentang *Abstract Data Type* serta meningkatkan pengetahuan mahasiswa tentang bahasa pemrograman C. Tugas besar ini juga sangat bermanfaat karena dapat melatih pola pikir mahasiswa saat bertemu dengan persoalan yang lumayan kompleks. Oleh karena itu, pemberian tugas besar ini banyak memberikan manfaat bagi mahasiswa.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 Game Treasure Hunt

Game Treasure Hunt adalah sebuah permainan tempat keberuntungan pemain diuji. Tujuan dari permainan ini adalah mengumpulkan “emas” sebanyak-banyaknya yang kemudian dikonversi menjadi poin. Pada awal permainan, tiap pemain akan memiliki tiga nyawa. Pemain akan dihadapkan dengan pilihan dimana pilihan tersebut memiliki 3 kemungkinan. Terdapat tiga kasus *node* berbeda yaitu *trap*, *treasure*, dan kosong. Bila pemain memilih *node* yang berisi *trap* maka, nyawa pemain akan dikurang satu. Bila pemain memilih *node* yang berisi *treasure* maka, pemain mendapat tambahan poin sebesar 1000. Bila pemain memilih *node* yang kosong maka, pemain harus menjawab pertanyaan. Bila pemain dapat menjawab pertanyaan dengan tepat maka, pemain mendapat tambahan poin sebesar 100 dan bila salah maka, poin pemain akan dikurang 10. Bila pada akhir permainan, pemain masih memiliki nyawa maka, untuk tiap nyawa akan dikonversi menjadi 3000 poin.

2.2 Perhitungan Skor Game Tower of Hanoi

Pada permainan Tower of Hanoi ditambahkan kalkulasi untuk skor akhir. Setelah pemain berhasil menyusun piringan pada tiang paling kanan (tiang C) maka, poin akhir adalah akumulasi jumlah piring dikalikan dua. Bila, langkah yang diperlukan pemain lebih dari 2 pangkat jumlah piring lalu dikurangi satu maka, poin akan dikurangi satu.

3 Struktur Data (ADT)

BNMO merupakan robot yang cukup kompleks dari segi algoritmanya karena diperlukan beberapa ADT agar *robot game console* BNMO dapat berjalan dengan lancar. Pada BNMO yang dimodifikasi untuk Tugas Besar 2, dibutuhkan 5 ADT, yaitu ADT *stackGame*, ADT *binTree*, ADT *mapGame*, ADT *linkedlistPoint*, dan ADT *stackToH*.

3.1 ADT stackGame

- Sketsa struktur data
Pada ADT *stackGame*, terdapat struktur data *stack* yang menyimpan elemen *stack* berupa infostack serta TOP yang menyimpan *address* dari sebuah *stack*. Prototype pada ADT *stackGame* berupa konstruktor untuk membuat *stackGame* yang kosong. Terdapat juga fungsi untuk mengecek apakah *stack* kosong atau *full*, serta fungsi untuk menambahkan atau menghapus sebuah elemen dari *stack*.
- Persoalan yang diselesaikan
ADT *stackGame* digunakan pada fungsi history dan reset history yang terdapat pada program BNMO
- Alasan pemilihan

Alasan pemilihan stackGame adalah pada fungsi history, urutan teratas merupakan permainan terakhir yang dimainkan. ADT stackGame juga digunakan pada fungsi reset history untuk menghapus semua *history* permainan yang dimainkan.

- Implementasi
Diimplementasikan sebagai ADT stackGame dengan nama *file header* “*stackGame.h*”

3.2 ADT binTree

- Sketa struktur data
Pada ADT binTree, terdapat struktur data EltypeT yang terdiri dari random dan *isTreasure* yang merupakan sebuah *integer*. Terdapat juga struktur data *TreeNode* yang terdiri dari infoT yang merupakan ElTypeT, left dan right yang merupakan AddressT. Pada ADT binTree, terdapat konstruktor berupa NewTree dan CreateTree untuk menghasilkan sebuah pohon biner yang baru. Terdapat juga fungsi lain seperti newTreeNode, deallocTreeNode, isEmpty, isOneElmt, isUnerLeft, isUnerRight, dan isBinary yang dibutuhkan untuk kepentingan program.
- Persoalan yang diselesaikan
ADT binTree digunakan pada game Treasure Hunt
- Alasan pemilihan
Alasan pemilihan ADT binTree pada game Treasure Hunt adalah untuk mempermudah jalannya program karena game Treasure Hunt merupakan game yang meminta pemain untuk bergerak ke kiri/kanan dan pemain bisa mendapatkan *treasure* atau jebakan atau tidak keduanya pada tiap langkah.
- Implementasi
Diimplementasikan untuk game Treasure Hunt, men-*generate binary tree* yang seimbang untuk keperluan game Treasure Hunt.

3.3 ADT mapGame

- Sketa struktur data
Pada ADT mapGame, terdapat struktur data infotype yang menyimpan elemen *key* berupa keytype serta value yang merupakan valuetype. Selain itu, juga terdapat struktur data map yang menyimpan elemen *map* berupa infotype serta count yang merupakan sebuah *address*. Prototype pada ADT mapGame berupa konstruktor untuk membuat *map* yang kosong. Terdapat juga primitif lain seperti IsEmptymap, IsFullmap, Valuemap, Insertmap, Deletemap, Ismembermap, Searchidxkey, sortmap, copymap yang dibutuhkan untuk kepentingan program.
- Persoalan yang diselesaikan
ADT mapGame digunakan pada fungsi readConfigStart, readSacefile, creategame, playgame, dan fungsi scoreboard (fungsi utama game).
- Alasan pemilihan
Alasan pemilihan mapGame adalah untuk memudahkan penyimpanan nama yang unik dan skor game pada fungsi scoreboard.
- Implementasi
Diimplementasikan sebagai ADT mapGame dengan nama *file header* “*mapGame.h*”

3.4 ADT *linkedlistPoint*

- Sketsa struktur data
Pada ADT *linkedlistPoint*, terdapat struktur data *ElmtList* yang terdiri dari info yang merupakan infolist serta *next* yang merupakan addressl. Selain itu, terdapat juga struktur data *List* yang terdiri dari *First* yang merupakan sebuah addressl. Terdapat prototype berupa konstruktor untuk membuat *linked list* yang kosong. Selain itu, terdapat beberapa primitif seperti *IsEmptyl*, *CreateEmptyl*, *Alokasil*, *Dealokasil*, *searchl*, dan masih banyak lagi yang dibutuhkan untuk kepentingan program.
- Persoalan yang diselesaikan
ADT *linkedlistPoint* digunakan pada game *Snake on Meteor*
- Alasan pemilihan
Alasan pemilihan ADT *linkedlistPoint* pada game *Snake on Meteor* adalah untuk mempermudah jalannya program
- Implementasi
Diimplementasikan sebagai ADT *linkedlistPoint* dengan nama *file header* "*linkedlistPoint.h*" untuk *generate snake* pada game *Snake on Meteor*.

3.5 ADT *stackTOH*

- Sketsa struktur data
Pada ADT *stackTOH*, terdapat struktur data *StackTOH* yang menyimpan elemen *stackTOH* berupa *infostackToH* serta *TOP* yang merupakan *addressStackToH*. Terdapat juga konstruktor untuk membuat *stackToH* yang kosong serta primitif untuk mengecek apakah *stack* kosong/full. Terdapat juga *PushStackToH* dan *PopStackToH* untuk menambahkan dan menghapus elemen *stack*.
- Persoalan yang diselesaikan
ADT *stackToH* diperlukan agar game *Tower of Hanoi* dapat berjalan dengan lancar
- Alasan pemilihan
Alasan pemilihan ADT *stackToH* pada game *Tower of Hanoi* adalah untuk mempermudah jalannya program karena *Tower of Hanoi* merupakan game dimana pemain harus memindahkan piringan ke sebuah tiang dan penggunaan *stackToH* merupakan ADT yang paling sesuai dengan peraturan permainan *Tower of Hanoi*
- Implementasi
Diimplementasikan sebagai ADT *stackTOH* dengan nama *file header* "*stackTOH.h*"

4 Program Utama

4.1 Main Menu

Ketika program pertama kali dijalankan, BNMO akan memperlihatkan main menu yang berisi *welcome page* dan empat menu pilihan yaitu START, LOAD, HELP, dan QUIT. Pada file main program, program akan menggunakan prosedur *display_welcoming* dan *mainmenu* untuk menampilkan pesan selamat datang dan *main menu*. Saat prosedur tersebut dipanggil, program akan meminta *input command* kepada *user*. Apabila pemain memasukkan *command* selain START, LOAD, HELP, dan QUIT, program akan menampilkan pesan kesalahan dan meminta user untuk memasukkan *command* lagi.

Command START digunakan jika pemain baru pertama kali memainkan BNMO atau me-restart permainan. *Command* LOAD <savefile> digunakan jika user sudah pernah memainkan BNMO. *Command* HELP di *main menu* memberikan panduan mengenai empat *command* yang tersedia. *Command* QUIT memungkinkan pemain untuk keluar dari program. Prosedur *scanParserStartStr* akan membaca input dari pengguna berupa pita karakter yang ditranslasikan menjadi string dan mengubah nilai string pada parameter *input-output* prosedur.

4.2 Pembacaan dan Inisialisasi File Konfigurasi

Pada program yang kami buat, kami memutuskan untuk membuat dua buah prosedur pembacaan file yaitu prosedur *readConfig* dan *readSaveFile*. Prosedur *readConfig* digunakan saat pemain memasukkan *command* START. Program akan langsung membaca file konfigurasi awal (*config.txt*) pada folder data yang berisikan daftar permainan. Daftar permainan pada file konfigurasi awal tersebut akan dibaca tiap barisnya dan akan dimasukkan kedalam array yang berisikan daftar permainan. Kemudian, array yang telah terisi tersebut akan ditampilkan ke layar.

Sedikit berbeda dengan prosedur *readConfig* yang langsung membaca file *config.txt*, prosedur *readSaveFile* akan membaca file hasil input dan melakukan algoritma yang sama dengan prosedur *readConfig*.

4.3 Pembacaan Command

Setelah pemain melakukan START atau LOAD, maka program akan menampilkan daftar *command* yang tersedia yang bisa dipilih user dengan fungsinya masing-masing. Setelah itu, program akan meminta masukan dari pemain. Apabila masukan pemain tidak valid (*command* tidak dikenali) maka, program akan meminta masukan ulang hingga masukan valid. Pada pembacaan *command*, kami menggunakan prosedur *scanParserStr* dengan meng-include ADT *mesin_input*. Pada prosedur *scanParserStr*, program akan meminta input dari pemain dengan menggunakan prosedur *STARTINPUT2*, lalu program akan memanggil fungsi *isSameString* untuk melakukan perbandingan antara input pemain dengan nama *command*, jika True maka program akan menjalankan prosedur sesuai dengan *command* terkait.

4.4 Command Dasar

Terdapat sembilan *command* dasar yaitu “CREATE GAME”, “LIST GAME”, “DELETE GAME”, “QUEUE GAME”, “PLAY GAME”, “SKIP GAME <n>”, “SAVE <namafile>”, “HELP”, dan “QUIT”. *Command* “CREATE GAME” digunakan untuk memungkinkan pengguna menambahkan permainan baru. *Command* “LIST GAME” adalah perintah yang digunakan untuk melihat daftar permainan. *Command* “DELETE GAME” adalah *command* yang digunakan untuk menghapus sebuah permainan dari List Game. *Command* “QUEUE GAME” adalah perintah yang dapat digunakan untuk menambahkan permainan ke antrian. *Command* “PLAY GAME” adalah *command* yang digunakan untuk memainkan permainan yang ada di antrian teratas. *Command* “SKIP GAME <n>” adalah perintah untuk melewati atau menghapus sejumlah <n> permainan teratas pada antrian. *Command* “SAVE <namafile>” adalah perintah yang digunakan untuk menyimpan state permainan. *Command* “HELP” adalah perintah yang dapat digunakan untuk mendapatkan keterangan mengenai *command* atau fitur yang tersedia. *Command* “QUIT” adalah perintah yang digunakan untuk keluar dari program.

4.5 Command “SCOREBOARD”

Command SCOREBOARD adalah perintah yang digunakan untuk melihat nama dan skor yang diperoleh pemain untuk semua game. *Command* SCOREBOARD memanfaatkan ADT mapGame yang merupakan implementasi ADT Map. Untuk menggabungkan tiap-tiap scoreboard dari masing-masing permainan digunakan ADT linked list dari nama permainan dan map dari scoreboard.

4.6 Command “RESET SCOREBOARD”

Command RESET SCOREBOARD adalah perintah yang dapat digunakan untuk melakukan *reset* atau penghapusan seluruh informasi pada scoreboard permainan. Ketika program menerima input untuk menjalankan *command* ini, pemain akan diberikan pilihan menghapus semua atau scoreboard salah satu permainan saja. Sesuai dengan pilihan pemain maka, map dari scoreboard akan dikosongkan.

4.7 Command “HISTORY <n>”

Command HISTORY <n> adalah perintah yang digunakan untuk melihat permainan yang telah dimainkan dengan <n> adalah jumlah permainan urutan teratas yang ingin ditampilkan. Perintah ini menggunakan ADT stackGame yang merupakan implementasi dari ADT Stack.

4.8 Command “RESETHISTORY”

Command RESETHISTORY adalah perintah yang digunakan untuk menghapus semua riwayat permainan yang telah dimainkan. Program akan melakukan pop terhadap isi stack pada *history* secara rekursif sampai *empty stack*.

4.9 Game Hangman

Game Hangman adalah permainan menebak kata yang merupakan nama negara di dunia dengan menggunakan ADT listGame yang merupakan implementasi ADT List. Di awal permainan, pemain bisa memilih untuk memulai permainan atau menambahkan kata dalam kamus tebakan. Bila pemain memilih untuk memulai permainan maka, pemain akan diberikan 10 kesempatan untuk melakukan tebakan. Tiap kali tebakan pemain tidak tepat maka, kesempatan akan dikurangi satu dan bila tepat maka, skor pemain akan ditambahkan satu. Program akan membaca file country.txt yang merupakan file berisi list nama negara sebagai kamus tebakan. Program akan men-*generate* suatu angka random untuk memilih kata yang akan ditebak. Program kemudian menunjukkan panjang kata tebakan dan meminta input tebakan dari pemain. Bila, pemain memilih menambahkan kata ke kamus tebakan maka, program akan mengakses file country.txt dan menambahkan input kata dari pemain.

4.10 Game Snake on Meteor

Game Snake on Meteor adalah permainan *snake* pada umumnya namun, dipersulit dengan kehadiran meteor. Pada awal permainan program akan me-*generate* peta, kepala dan badan *snake*, serta makanan. Program memanfaatkan ADT *linkedlistPoint* yang merupakan implementasi ADT *linked list*. *Snake* terdiri dari dua bagian yaitu kepala yang ditandai dengan H dan pada awal permainan 2 yang ditandai dengan angka (1,2, dst.) Makanan ditandai dengan O sedangkan, meteor dengan M dan *obstacle* dengan X. Pemain kemudian diminta memasukkan perintah untuk menggerakkan ular dengan ketentuan ASDW yaitu A ke kiri, D ke kanan, S ke bawah dan W ke atas. Pemain juga tidak bisa menggerakkan *snake*-nya mengenai meteor.

Peta yang digunakan merupakan implementasi logika matriks. Peta berukuran 5 x 5 dengan total kotak pergerakan 25. Tiap kotak diidentifikasi dengan koordinat (x,y).

Untuk tiap input pergerakan maka, program akan me-*generate* random number untuk lokasi meteor dan makanan. *Random number* dijadikan sebuah koordinat (x,y) untuk menentukan letak *random* meteor. Hal serupa juga dilakukan terhadap lokasi makanan tiap kali *snake* pemain berhasil memakan makanan.

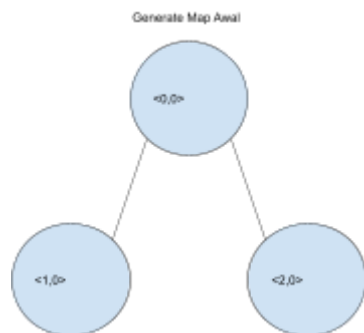
Inti dari permainan adalah memakan makanan sebanyak-banyaknya untuk memanjangkan tubuh *snake* selagi menghindari *obstacle*, meteor, tepi peta dan memastikan pergerakan tidak menabrak badan *snake* sendiri. Bila pemain berhasil memakan makanan maka, badan *snake* akan bertambah dengan munculnya angka tambahan. Bila badan *snake* terkena meteor maka, bagian badan tersebut akan seakan-akan “hilang” dan badan *snake* akan dikurangi. Bila kepala *snake* yang terkena meteor maka, permainan selesai. Bila *snake* mengenai *obstacle* atau tepi peta maka, permainan selesai. Bila, kepala *snake* mengenai badannya sendiri maka, permainan selesai. Poin akhir pemain dihitung dari panjang badan *snake* dan kasus penyebab permainan berakhir. Secara umum, panjang badan *snake* (termasuk kepala, badan, ekor) dikalikan dua namun, bila kasus penyebabnya adalah kepala *snake* terkena meteor maka, kepala *snake* tidak dihitung.

4.11 Game Tower of Hanoi

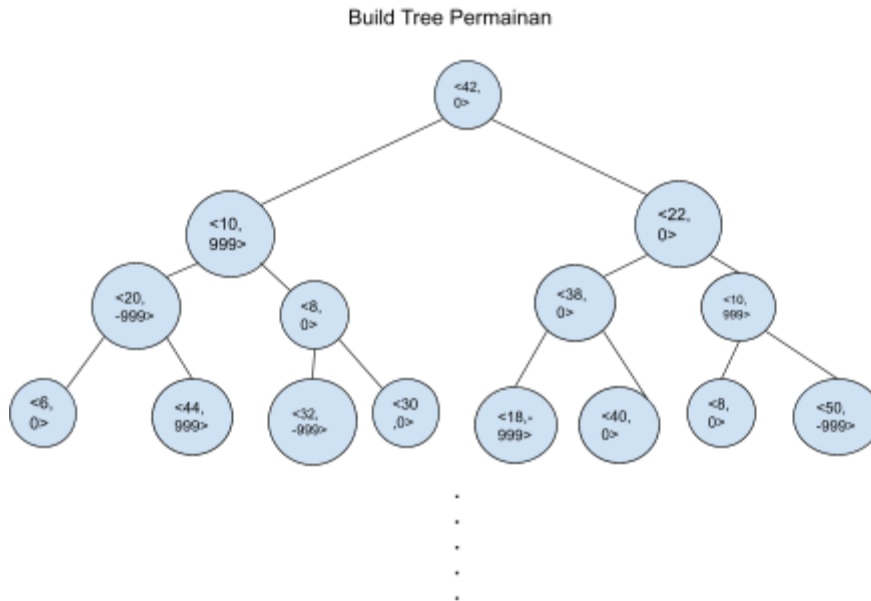
Game Tower of Hanoi adalah permainan dimana pemain harus menyusun piringan dari yang terbesar ke terkecil dari salah satu tiang ke tiang paling kanan. Pada awal permainan permainan meminta input jumlah piringan. Permainan ini menggunakan logika struktur data *stack* dengan memanfaatkan ADT StackTOH yang merupakan implementasi ADT *Stack*. Terdapat tiga tiang atau tower yaitu tower A (paling kiri dan merupakan tiang *default* asal), tower B (tiang tengah), dan tower C (paling kanan dan merupakan tujuan akhir). Untuk memindahkan piringan, pemain dapat memasukkan nama tiang asal dan nama tiang tujuan. Bila pemain memindahkan piringan yang tidak lebih kecil dari piringan yang sudah ada di tower paling kanan maka, program akan menunjukkan pesan “Gagal memindahkan piringan karena piring asal lebih besar!”. Pemain dapat memanfaatkan tower B untuk menyusun. Setelah berhasil maka, poin akhir adalah akumulasi jumlah piring dikalikan dua. Bila, langkah yang diperlukan pemain lebih dari 2 pangkat jumlah piring lalu dikurangi satu maka, poin akan dikurangi satu.

4.12 Game ADT TREE “Treasure Hunt”

Game Treasure Hunt adalah sebuah permainan tempat keberuntungan pemain diuji dengan tujuan mengumpulkan “emas” sebanyak-banyaknya yang kemudian dikonversi. Permainan memanfaatkan ADT *binary tree* yang *node*-nya berstruktur $\langle \text{random integer, isTreasure} \rangle$. Pada awal permainan, pemain diminta untuk memilih *path adventure treasure hunt*, dalam hal ini dibuat *tree* sebagai map awal yang *left node*-nya adalah angka *random* 1-2 (tidak boleh sama). *Tree* awal ini dibuat untuk menentukan *path* pemain (1 untuk tantangan dengan soal pengetahuan umum dan 2 untuk tantangan dengan soal *logical arithmetic*).



Setelah itu, *build map 2* dengan *binary tree* dengan kedalaman *tree* yang dapat diubah-ubah pada file header (saat ini kedalaman *tree* di-set 10, sehingga ada 2 pangkat 10 *node*). Pada saat *build tree* permainan secara rekursif, dilakukan randomisasi angka pada tiap *node*. *Random number* yang didapatkan menunjukkan kode soal dan *node.istreasure* menunjukkan tiga macam kode yaitu kode treasure (999), kode trap (-999), kode tidak ada apa-apa atau *node* kosong(0).



Setelah *build tree* utama, pemain dapat mengakses *tree* dengan mengetikkan “LEFT” atau “RIGHT”, setelah itu program akan memeriksa `node.istreasure` terlebih dahulu. Jika `node.treasure` bernilai 999 (*treasure*) maka poin akan ditambah 1000, jika `node.treasure` bernilai -999 (*trap*) maka nyawa pemain akan berkurang 1 dan jika `node.istreasure` bernilai 0 maka akan dilakukan pengecekan kode soal (`node.random`). Jika `node.istreasure` bernilai 0, pemain diberikan pertanyaan sesuai *path* yang dipilih di awal permainan (jika mendapat *path* pengetahuan umum maka akan ada pengaksesan di *array of* kalimat hasil pembacaan file data yang berisi soal dan jawaban). Pemain harus menjawab pertanyaan tersebut dengan benar. Jika menjawab salah maka poin akan berkurang 10 dan jika berhasil menjawab benar maka poin akan bertambah 100. Jika pemain kurang beruntung dan terkena trap 3 kali, nyawa akan habis dan *game over*.

5 Algoritma-Algoritma Menarik

5.1 Build Balance Tree dengan Random Number Tiap Node

Pada permainan Treasure Hunt sebagai implementasi ADT *Binary Tree*, dibuat pohon dengan kedalaman x (x berarti banyak *turn* yang diinginkan). *Tree* yang digunakan adalah *balance tree*, supaya tidak ada kasus pengaksesan elemen dengan address NULL sebelum x *turn* berlangsung (kecuali saat *game over*) padahal *tree* sebelumnya masih memiliki banyak *node*. Dengan *balance tree*, jika kita menginginkan *tree* dengan kedalaman x , maka kita memerlukan *tree* dengan jumlah *node* 2^x .

Pada saat *build balance tree* kita juga memerlukan randomisasi angka setiap *node* (angka random tidak boleh cenderung sama dengan antar satu *node* dengan *node* lain). Dengan fungsi `srand(time(NULL))` dan `rand()` program akan merandom angka sesuai dengan waktu program dijalankan, namun program bekerja dengan sangat cepat (kurang dari satu detik) sehingga hasil angka random pada semua *node* akan sama, karena waktu

program diakses juga sama. Penanganan pertama yang telah dicoba untuk menangani kasus ini adalah dengan penambahan fungsi delay(1) setiap akan menjalankan srand(time(NULL)) saat *generate 1 node* secara rekursif. Akan tetapi, ditemukan bahwa penambahan delay(1) ini kurang efisien karena dengan *balance tree*, jika kita ingin membuat *tree* dengan kedalaman 10 maka kita memerlukan 2^{10} atau 1024 *node*, dengan delay(1) maka program akan berjalan dengan sangat lama karena setiap *generate node* dengan random number program memerlukan 1 detik hanya karena ingin mendapatkan angka-angka random yang tidak sama antar-*node*, padahal program bisa berjalan lebih cepat (hanya karena permasalahan random angka).

Dalam menyelesaikan masalah ini, ditemukan algoritma yang lebih efisien waktu daripada menggunakan fungsi delay(1). Dengan fungsi rekursif yang dibuat untuk membuild *balance tree*, disadari bahwa *n* akan berkurang seiring pembuatan cabang-cabang yang baru (*n* selalu berubah). Sehingga, kita bisa menggunakan nilai *n* (rand() + *n*) supaya random angka yang dihasilkan tiap *node* tidak sama semua karena *n* selalu berbeda. Implementasi algoritma dapat dilihat pada tabel berikut.

```
BinTree BuildMap(int rentangrandom, int n){
    int infotreasure[]={999,0,999,0,0,-999,0,0,-999,0,999,0,0,-999};
    srand(time(NULL));
    AddressT P;
    int X,Y;
    BinTree R, L;
    int nL, nR;
    if (n == 0){
        return Nil;
    } else{
        X = (rand() + n) % rentangrandom +1;
        // ... suatu operasi
        ElTypeT node;
        node.random=X;
        Y = rand() % 14;
        Y = (Y+n+3) % 14;
        node.isTreasure=infotreasure[Y];
        P = newTreeNode(node);
        if (P != Nil){
            ROOT(P) = node;
            nL = n / 2;
            nR = n - nL - 1;
            L = BuildMap(rentangrandom, nL);
            R = BuildMap(rentangrandom, nR);
            LEFT(P) = L;
            RIGHT(P) = R; }
        return P;
    }
}
```

6 Data Test

6.1 Data Test 1

- **Fitur yang dites**

Tes ini dilakukan untuk memeriksa apakah program dapat menampilkan nama dan skor untuk semua game yang terdaftar. Scoreboard juga diurutkan berdasarkan skor

- **Hasil yang diharapkan**

```
// 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          |
|-----|
| BNMO          | 80            |
| Finn          | 62            |

**** SCOREBOARD GAME SNAKE ON METEOR****
| NAMA          | SKOR          |
|-----|
| BNMO          | 13            |
| Finn          | 11            |

**** SCOREBOARD GAME RNG****
| NAMA          | SKOR          |
|-----|
| BNMO          | 99            |
| Finn          | 1             |

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

- Hasil yang ditampilkan

Berikut adalah scoreboard BNMO!	
1. RNG	
NAMA	SKOR
Doni	300
Indra	300
2. Diner DASH	
NAMA	SKOR
-----SCOREBOARD KOSONG-----	
3. HANGMAN	
NAMA	SKOR
-----SCOREBOARD KOSONG-----	
4. TOWER OF HANOI	
NAMA	SKOR
-----SCOREBOARD KOSONG-----	
5. SNAKE ON METEOR	
NAMA	SKOR
Indra	10
Freak	6
6. STI Mencari JODOH	
NAMA	SKOR
Doni	89
7. TREASURE HUNT	
NAMA	SKOR
Naura	1000
BNMO	990
8. MINECRAFT	
NAMA	SKOR
-----SCOREBOARD KOSONG-----	
ENTER COMMAND: █	

6.2 Data Test 2

- Fitur yang dites

Tes ini dilakukan untuk memeriksa apakah program dapat melakukan reset terhadap scoreboard permainan, namun kasusnya jika skor semua game ingin direset

- Hasil yang diharapkan

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.

- Hasil yang ditampilkan

```

-----
ENTER COMMAND: RESET SCOREBOARD
DAFTAR SCOREBOARD:
0. ALL
1. RNG
2. Diner DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. STI Mencari JODOH
7. TREASURE HUNT
8. MINECRAFT

SCOREBOARD YANG INGIN DIHAPUS: 0
APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD ALL? (YA/TIDAK) YA

Scoreboard berhasil di-reset.
-----

```

```

Berikut adalah scoreboard BNMO!

1. RNG
|-----|
| NAMA      | SKOR    |
|-----|
|-----SCOREBOARD KOSONG-----|

2. Diner DASH
|-----|
| NAMA      | SKOR    |
|-----|
|-----SCOREBOARD KOSONG-----|

3. HANGMAN
|-----|
| NAMA      | SKOR    |
|-----|
|-----SCOREBOARD KOSONG-----|

4. TOWER OF HANOI
|-----|
| NAMA      | SKOR    |
|-----|
|-----SCOREBOARD KOSONG-----|

```

```

5. SNAKE ON METEOR
|-----|
| NAMA      | SKOR    |
|-----|
|-----SCOREBOARD KOSONG-----|

6. STI Mencari JODOH
|-----|
| NAMA      | SKOR    |
|-----|
|-----SCOREBOARD KOSONG-----|

7. TREASURE HUNT
|-----|
| NAMA      | SKOR    |
|-----|
|-----SCOREBOARD KOSONG-----|

8. MINECRAFT
|-----|
| NAMA      | SKOR    |
|-----|
|-----SCOREBOARD KOSONG-----|

```

6.3 Data Test 3

- Fitur yang dites

Tes ini dilakukan untuk memeriksa apakah program dapat melakukan reset terhadap scoreboard permainan, namun kasusnya jika hanya skor dari salah satu game yang ingin dihapus

- Hasil yang diharapkan

```

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.

- Hasil yang ditampilkan

```
-----
ENTER COMMAND: RESET SCOREBOARD
DAFTAR SCOREBOARD:
0. ALL
1. RNG
2. Diner DASH
3. HANGMAN
4. TOWER OF HANOI
5. SNAKE ON METEOR
6. STI Mencari JODOH
7. TREASURE HUNT
8. MINECRAFT

SCOREBOARD YANG INGIN DIHAPUS: 6
APAKAH KAMU YAKIN INGIN MELAKUKAN RESET SCOREBOARD STI Mencari JODOH? (YA/TIDAK) YA

Scoreboard berhasil di-reset.
-----
```

```
5. SNAKE ON METEOR
|-----|
| NAMA      | SKOR |
|-----+-----|
| Indra      | 10   |
| Freak     | 6    |
|-----+-----|

6. STI Mencari JODOH
|-----|
| NAMA      | SKOR |
|-----+-----|
|-----SCOREBOARD KOSONG-----|

7. TREASURE HUNT
|-----|
| NAMA      | SKOR |
|-----+-----|
| Naura     | 1000 |
| BNMO      | 990  |
|-----+-----|
```

6.4 Data Test 4

- Fitur yang dites

Tes ini dilakukan untuk melihat permainan apa saja yang telah dimainkan dari data yang sudah ada dari file konfigurasi, kondisinya jika <n> lebih kecil dari jumlah permainan yang telah dimainkan

- Hasil yang diharapkan

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

- Hasil yang ditampilkan

```
-----
ENTER COMMAND: HISTORY 2
Berikut adalah daftar game yang telah dimainkan
1. SNAKE ON METEOR
2. RNG
-----
```

6.5 Data Test 5

- Fitur yang dites

Tes ini dilakukan untuk melihat permainan apa saja yang telah dimainkan dari data yang sudah ada di file konfigurasi, namun kondisinya jika <n> lebih besar dari jumlah permainan yang telah dimainkan

- Hasil yang diharapkan

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

- Hasil yang ditampilkan

```
ENTER COMMAND: HISTORY 9
Berikut adalah daftar game yang telah dimainkan
1. SNAKE ON METEOR
2. RNG
3. RNG
4. STI MENCARI JODOH
5. SNAKE ON METEOR
-----
```

6.6 Data Test 6

- Fitur yang dites

Tes ini dilakukan untuk memeriksa apakah program dapat menghapus semua history permainan yang dimainkan

- Hasil yang diharapkan

```
ENTER COMMAND: RESET HISTORY

APAKAH KAMU YAKIN INGIN MELAKUKAN RESET HISTORY? YA

History berhasil di-reset.
```

- Hasil yang ditampilkan

```
-----
ENTER COMMAND: HISTORY 1
Berikut adalah daftar game yang telah dimainkan
1. RNG
-----
ENTER COMMAND: RESET HISTORY
Apakah kamu yakin ingin melakukan reset history? (YA/TIDAK) YA

History berhasil di-reset.
-----
ENTER COMMAND: HISTORY 1
Daftar history kosong.
-----
```

6.7 Data Test 7

- Fitur yang dites

Tes ini dilakukan untuk memeriksa apakah program dapat membatalkan command history yang dimasukan oleh user

- Hasil yang diharapkan

```
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
```

- Hasil yang ditampilkan

```
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
1. RNG
-----
```

6.8 Data Test 8

- Fitur yang dites

Tes ini dilakukan untuk memeriksa apakah game hangman dapat dimulai dengan lancar

- Hasil yang diharapkan

Game hangman sudah dapat dimainkan

- Hasil yang ditampilkan

```

ENTER COMMAND: PLAY GAME
Berikut adalah daftar antrian game-mu
1. HANGMAN

Loading HANGMAN . . . .

Selamat Datang di Permainan Hangman
Edisi : World's Country!

Pilih menu di bawah ini:
1. Mulai Permainan
2. Tambahkan Negara Baru

Masukkan pilihan menu: 1
Tebakan sebelumnya: -
Kata: _ _ _ _ _
Kesempatan: 10
Masukkan tebakan: |

```

6.9 Data Test 9

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa *output* yang dikeluarkan program jika tebakan yang dimasukan salah
- **Hasil yang diharapkan**
Kesempatan bermain berkurang, program akan meminta pemain untuk memasukan huruf lain
- Hasil yang ditampilkan

```

Masukkan pilihan menu: 1
Tebakan sebelumnya: -
Kata: _ _ _ _ _
Kesempatan: 10
Masukkan tebakan: Z

Tebakan sebelumnya: z
Kata: _ _ _ _ _
Kesempatan: 9
Masukkan tebakan: |

```

6.10 Data Test 10

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa *output* yang dikeluarkan program jika tebakan yang dimasukan benar

- **Hasil yang diharapkan**
Kesempatan bermain tidak berkurang, program akan meminta pemain untuk memasukkan huruf lain
- **Hasil yang ditampilkan**

```
Tebakan sebelumnya: z
Kata: _ _ _ _ _
Kesempatan: 9
Masukkan tebakan: A

Tebakan sebelumnya: za
Kata: _ _ _ _ A _
Kesempatan: 9
Masukkan tebakan: |
```

6.11 Data Test 11

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa *output* yang dikeluarkan program jika semua huruf berhasil ditebak
- **Hasil yang diharapkan**
Akan keluar teks yang menandakan bahwa kata sudah berhasil ditebak
- **Hasil yang ditampilkan**

```
Tebakan sebelumnya: zafreuonw
Kata: N O R W A _
Kesempatan: 6
Masukkan tebakan: Y

Berhasil menebak kata NORWAY! Kamu mendapatkan 6 poin!

Tebakan sebelumnya: -
Kata: _ _ _ _ _
Kesempatan: 6
Masukkan tebakan: |
```

6.12 Data Test 12

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa apakah game Tower of Hanoi dapat dimulai dengan lancar
- **Hasil yang diharapkan**
Game Tower of Hanoi dapat mulai dimainkan

- Hasil yang ditampilkan

```

ENTER COMMAND: PLAY GAME
Berikut adalah daftar antrian game-mu
1. TOWER OF HANOI

Loading TOWER OF HANOI . . . .

Masukkan jumlah piringan: 

```

6.13 Data Test 13

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa *output* yang dikeluarkan program jika piringan berhasil dipindah
- **Hasil yang diharapkan**
Piringan berhasil dipindah ke *tower* lain
- Hasil yang ditampilkan

```

TIANG ASAL: A
TIANGTUJUAN: C

Berhasil memindahkan piringan!

  |   |   |
*** |   |
***** | *
--- |   |
  A   B   C

```

6.14 Data Test 14

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa *output* yang dikeluarkan program jika piringan gagal dipindah
- **Hasil yang diharapkan**
Piringan gagal dipindah ke *tower* lain karena piring asal lebih besar
- Hasil yang ditampilkan

```

  |   |   |
*** |   |
***** | *
--- |   |
  A   B   C
TIANG ASAL: A
TIANGTUJUAN: C

Gagal memindahkan piringan karena piring asal lebih besar!

```

6.15 Data Test 15

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa *output* program jika game Tower of Hanoi sudah selesai dimainkan
- **Hasil yang diharapkan**
Game selesai dimainkan, terdapat *output* berupa skor game. User juga diminta untuk memasukan nama
- **Hasil yang ditampilkan**

```
| | *
| | ***
| | *****
---
A B C
Kamu berhasil!

Skor didapatkan: 6

Terima kasih telah bermain TOWER OF HANOI!
-----
Skor akhir: 6
Nama:Test
```

6.16 Data Test 16

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa apakah game Snake on Meteor dapat dimulai dengan lancar
- **Hasil yang diharapkan**
Game Snake on Meteor dapat mulai dimainkan
- **Hasil yang ditampilkan**

```
ENTER COMMAND: PLAY GAME
Berikut adalah daftar antrian game-mu
1. SNAKE ON METEOR

Loading SNAKE ON METEOR . . . .

Selamat datang di Snake on Meteor!

Mengenerate peta, snake, dan makanan. . . .

Berikut merupakan peta permainan:
+---+---+---+---+---+
| 0 | | | | | |
+---+---+---+---+---+
| | | | | | X |
+---+---+---+---+---+
| | | X | | | |
+---+---+---+---+---+
| H | | | 2 | 1 |
+---+---+---+---+---+
| | | | | | |
+---+---+---+---+---+

Masukkan perintah: |
```

6.17 Data Test 17

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa apakah *snake* dapat berubah posisi sesuai dengan *command* yang dimasukan
- **Hasil yang diharapkan**
Snake berhasil berubah posisi sesuai *command* yang dimasukan pemain
- **Hasil yang ditampilkan**

```
Berikut merupakan peta permainan:
+ --- + --- + --- + --- +
| 0 | | | | |
+ --- + --- + --- + --- +
| | | | | X |
+ --- + --- + --- + --- +
| | | X | | |
+ --- + --- + --- + --- +
| H | | | 2 | 1 |
+ --- + --- + --- + --- +
| | | | | |
+ --- + --- + --- + --- +

Masukkan perintah: w
Berhasil bergerak!
Berikut merupakan peta permainan:
+ --- + --- + --- + --- +
| 0 | | | | |
+ --- + --- + --- + --- +
| | | M | | X |
+ --- + --- + --- + --- +
| H | | X | | |
+ --- + --- + --- + --- +
| 1 | | | | 2 |
+ --- + --- + --- + --- +
| | | | | |
+ --- + --- + --- + --- +

Masukkan perintah: |
```

6.18 Data Test 18

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa *output* yang dikeluarkan saat *snake* berhasil makan
- **Hasil yang diharapkan**
Snake berhasil makan dan bertambah panjang ekornya sebanyak satu unit
- **Hasil yang ditampilkan**

```
Berikut merupakan peta permainan:
+ --- + --- + --- + --- +
| 0 | | | | |
+ --- + --- + --- + --- +
| H | | | M | X |
+ --- + --- + --- + --- +
| 1 | | | X | | |
+ --- + --- + --- + --- +
| 2 | | | | |
+ --- + --- + --- + --- +
| | | | | |
+ --- + --- + --- + --- +

Masukkan perintah: w
Berhasil bergerak!
Berikut merupakan peta permainan:
+ --- + --- + --- + --- +
| H | | | | |
+ --- + --- + --- + --- +
| 1 | | | | X |
+ --- + --- + --- + --- +
| 2 | 0 | X | | |
+ --- + --- + --- + --- +
| 3 | | | | |
+ --- + --- + --- + --- +
| | | | M | |
+ --- + --- + --- + --- +

Masukkan perintah: |
```

6.19 Data Test 19

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa *output* yang dikeluarkan saat *snake* terkena meteor
- **Hasil yang diharapkan**
Panjang dari *snake* akan berkurang 1

- Hasil yang ditampilkan

```

Berikut merupakan peta permainan:
+ --- + --- + --- + --- + --- +
| X |   |   |   |   |   |
+ --- + --- + --- + --- + --- +
|   | H | 1 | 2 |   |   |
+ --- + --- + --- + --- + --- +
|   |   |   | 3 | M |   |
+ --- + --- + --- + --- + --- +
|   | 0 |   | 4 |   |   |
+ --- + --- + --- + --- + --- +
|   |   | X |   |   |   |
+ --- + --- + --- + --- + --- +

Masukkan perintah: a

Berhasil bergerak!
Berikut merupakan peta permainan:
+ --- + --- + --- + --- + --- +
| X |   |   |   |   |   |
+ --- + --- + --- + --- + --- +
| H | 1 | M | 3 |   |   |
+ --- + --- + --- + --- + --- +
|   |   |   | 4 |   |   |
+ --- + --- + --- + --- + --- +
|   | 0 |   |   |   |   |
+ --- + --- + --- + --- + --- +
|   |   | X |   |   |   |
+ --- + --- + --- + --- + --- +

Kamu terkena meteor!
Berikut merupakan peta permainan:
+ --- + --- + --- + --- + --- +
| X |   |   |   |   |   |
+ --- + --- + --- + --- + --- +
| H | 1 | M | 2 |   |   |
+ --- + --- + --- + --- + --- +
|   |   |   | 3 |   |   |
+ --- + --- + --- + --- + --- +
|   | 0 |   |   |   |   |
+ --- + --- + --- + --- + --- +
|   |   | X |   |   |   |
+ --- + --- + --- + --- + --- +

Masukkan perintah: |

```

6.20 Data Test 20

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa *output* yang dikeluarkan saat *snake* ingin pergi ke titik yang terkena meteor pada *turn* sebelumnya
- **Hasil yang diharapkan**
Pemain diminta untuk memasukan *command* lain karena titik tersebut masih panas

- Hasil yang ditampilkan

```

Berikut merupakan peta permainan:
+---+---+---+---+---+
| 1 | H | M |   |   |
+---+---+---+---+---+
| 2 |   |   |   | X |
+---+---+---+---+---+
| 3 | O | X |   |   |
+---+---+---+---+---+
|   |   |   |   |   |
+---+---+---+---+---+
|   |   |   |   |   |
+---+---+---+---+---+
|   |   |   |   |   |
+---+---+---+---+---+

Masukkan perintah: d

Berhasil bergerak!
Meteor masih panas! Anda belum dapat kembali ke titik tersebut. Silahkan masukkan command lainnya.
Berikut merupakan peta permainan:
+---+---+---+---+---+
| 1 | H | M |   |   |
+---+---+---+---+---+
| 2 |   |   |   | X |
+---+---+---+---+---+
| 3 | O | X |   |   |
+---+---+---+---+---+
|   |   |   |   |   |
+---+---+---+---+---+
|   |   |   |   |   |
+---+---+---+---+---+
|   |   |   |   |   |
+---+---+---+---+---+

Masukkan perintah: |

```

6.21 Data Test 21

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa *output* yang dikeluarkan saat *snake* melakukan pergerakan ke diri sendiri
- **Hasil yang diharapkan**
Pemain diminta untuk memasukan *command* lain karena *snake* tidak dapat bergerak ke tubuh sendiri
- Hasil yang ditampilkan

```

Berikut merupakan peta permainan:
+---+---+---+---+---+
| X |   |   |   |   |
+---+---+---+---+---+
|   |   |   |   | 4 |
+---+---+---+---+---+
|   |   |   | H | 3 |
+---+---+---+---+---+
|   | O |   | 1 | 2 |
+---+---+---+---+---+
| M |   |   | X |   |
+---+---+---+---+---+

Masukkan perintah: d

Berhasil bergerak!
Snake tidak bisa bergerak ke badan sendiri!!
Berikut merupakan peta permainan:
+---+---+---+---+---+
| X |   |   |   |   |
+---+---+---+---+---+
|   |   |   |   | 4 |
+---+---+---+---+---+
|   |   |   | H | 3 |
+---+---+---+---+---+
|   | O |   | 1 | 2 |
+---+---+---+---+---+
| M |   |   | X |   |
+---+---+---+---+---+

Masukkan perintah: |

```

6.22 Data Test 22

- **Fitur yang dites**
Tes ini dilakukan untuk memeriksa *output* yang dikeluarkan saat *snake* berhasil makan sekaligus terkena meteor pada kepala

- Hasil yang diharapkan
Snake berhasil bergerak namun permainan berakhir karena kepala *snake* terkena meteor
- Hasil yang ditampilkan



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 program dapat menampilkan scoreboard game	Memasukkan <i>command SCOREBOARD</i>	Data Test 1	Menampilkan tampilan scoreboard	Sesuai dengan hasil yang diharapkan
2.	RESET SCOREBOARD	Memeriksa apakah program dapat melakukan reset terhadap scoreboard permainan	Memasukkan <i>command RESET SCOREBOARD</i>	Data Test 2 Data Test 3	Melakukan reset pada scoreboard permainan	Sesuai dengan hasil yang diharapkan
3.	HISTORY	Memeriksa apakah program dapat menampilkan judul permainan yang sudah pernah dimainkan	Memasukkan <i>command HISTORY <n></i>	Data Test 4 Data Test 5	Program menampilkan judul game yang sudah pernah dimainkan	Sesuai dengan hasil yang diharapkan
4.	RESET HISTORY	Memeriksa apakah program dapat menghapus semua history permainan yang dimainkan	Memasukkan <i>command RESET HISTORY</i>	Data Test 6 Data Test 7	Program menghapus semua history permainan	Sesuai dengan hasil yang diharapkan
5.	Game <i>Hangman</i>	Memeriksa apakah game <i>hangman</i> dapat berjalan dengan lancar	Memasukkan <i>command PLAYGAME</i> , lalu memasukkan <i>command HANGMAN</i>	Data Test 8 Data Test 9 Data Test 10 Data Test 11	Program memainkan game <i>Hangman</i>	Sesuai dengan hasil yang diharapkan

6.	Game <i>Tower of Hanoi</i>	Memeriksa apakah game <i>Tower of Hanoi</i> dapat berjalan dengan lancar	Memasukkan <i>command</i> PLAYGAME lalu memasukkan <i>command</i> TOWER OF HANOI	Data Test 12 Data Test 13 Data Test 14 Data Test 15	Program memainkan game <i>Tower of Hanoi</i>	Sesuai dengan hasil yang diharapkan
7.	Game <i>Snake on meteor</i>	Memeriksa apakah game <i>Snake on Meteor</i> dapat berjalan dengan lancar	Memasukkan <i>command</i> PLAYGAME lalu memasukkan <i>command</i> SNAKE ON METEOR	Data Test 16 Data Test 17 Data Test 18 Data Test 19 Data Test 20 Data Test 21 Data Test 22	Program memainkan game <i>Snake on Meteor</i>	Sesuai dengan hasil yang diharapkan

8 Pembagian Kerja dalam Kelompok

Nama	Tugas
18221071/Ahmad Rizki	<ul style="list-style-type: none"> • Membuat main program • Membuat game Snake on Meteor • Revisi tubes 1 • Membuat game <i>Hangman</i>
18221081/Nadine Aliya	<ul style="list-style-type: none"> • Membuat fungsi <i>history, reset history</i> • Membuat laporan • Notul asistensi
18221099/Clara Alrosa Fernanda Sinaga	<ul style="list-style-type: none"> • Membuat game <i>Hangman</i> • Membuat laporan • Notul asistensi
18221111/Pramaditya Fajri M.	<ul style="list-style-type: none"> • Membuat game <i>Tower of Hanoi</i> • Membuat driver
18221173/Naura Valda Prameswari	<ul style="list-style-type: none"> • Membuat main program • Membuat game bonus <i>tree</i> • Membuat fungsi <i>scoreboard, reset scoreboard</i> • Revisi tubes 1

9 Lampiran

9.1 Deskripsi Tugas Besar 2

System Mechanics

1. About the System

BNMO merupakan suatu robot game console yang dapat menjalankan permainan.

BNMO memiliki beberapa fitur utama, yaitu:

- a. Memainkan game
- b. Menambahkan game
- c. Menghapus game
- d. Mengurutkan game yang akan dimainkan
- e. Menampilkan game yang telah dimainkan
- f. 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 *command* yang digunakan:

a. *Command* Dasar

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

b. SCOREBOARD

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

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
4. Konfigurasi Sistem
File konfigurasi akan dibaca saat memulai permainan. File ini menyimpan data-data yang disimpan ketika sistem dijalankan sebelumnya
 5. Spesifikasi game
 - a. 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.
 - b. 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).

c. 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:

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

6. Game tambahan

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

9.2 Notulen Rapat



Tanggal	Kesimpulan	Anggota yang terlibat
21 November 2022	<ul style="list-style-type: none">- Memahami spesifikasi tugas- Pembagian kerja untuk membuat <i>command</i> yang diperlukan.- Menghubungi asisten perihal asistensi 1- Pembagian kerja untuk membuat <i>command</i> yang diperlukan.	Semua Anggota
24 November 2022	<ul style="list-style-type: none">- Pengecekan progres tiap fitur- Membahas spek yang masih kurang dimengerti- Asistensi 1	Semua Anggota
30 November 2022	<ul style="list-style-type: none">- Mengecek program secara garis besar- Melengkapi laporan- Mengurus bug-bug yang masih ada pada program	Semua Anggota
1 Desember 2022	<ul style="list-style-type: none">- Finalisasi program- Menyelesaikan laporan- Asistensi 2	Semua Anggota

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

No. Kelompok/Kelas : 05/01
 Nama Kelompok : DoaWibu
 Anggota Kelompok (Nama/NIM) :
 1. Ahmad Rizki/18221071
 2. Nadine Aliya Putri/18221081
 3. Clara Alrosa Fernanda Sinaga/18221099
 4. Pramaditya Fajri Migfar/18221111
 5. Naura Valda Prameswari/18221173

Asisten Pembimbing : Jason Stanley Yoman

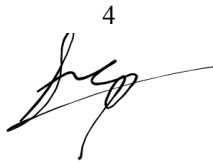
Asistensi I

Tanggal : Kamis, 24 November 2022	CATATAN ASISTENSI Nadine: Di function history harus ada game dari yang maintenance apa gausah kak? Kak Jason: Gausah, karena maintenance udah dihapus dari spek Clara: Game hangman gadijelasin harus pake ADT apa, kira2 ada ADT tertentu ga kak yang harus dipakai? Kak Jason: Terserah mau pakek apa, ga harus linked list, jadinya kalo mau array juga gapapa Clara: Di hangman besar point nya dibebasin apa gimana kak? Kak Jason: Bebas, yang penting kata yang lebih banyak skor nya harus lebih gede karena lebih susah ditebak Naura: Buat penambahan obstacle, obstacle nya di random waktu tiap kita generate game, tiap game baru itu harus random lagi apa gimana? Jason: Terserah, tapi yang penting obstacle nya bisa ditaro dimanapun (obstacle nya di setting di code nya aja) Amriz: Kalau badannya kena meteor, terus nyambunginnya setelah jalan satu kali apa gimana kak?
Tempat : Zoom Meeting	
Kehadiran Anggota Kelompok: No NIM Tanda tangan 1  Ahmad Rizki 18221071 2  Nadine Aliya Putri 18221081 3	

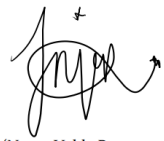


Clara Alrosa Fernanda Sinaga
18221099

4



Pramaditya Fajri Migfar
18221111



Naura Valda Prameswari
18221173

Jason:





Intinya ketika badannya kena meteor, otomatis badannya ilang. Nah itu untuk term selanjutnya ularnya jalan seperti biasa.



Tanda Tangan Asisten:



Jason Stanley Yoman
13519019

Asistensi II

Tanggal : 01/12/2022	<p>CATATAN ASISTENSI</p> <p>Kak Jason: Boleh ceritain progressnya udah sampai mana Amriz: Udah aman kak, tinggal laporan</p> <p>Nadine: Buat datatest di laporan itu harus ada bagian game nya apa cuman fugsi programnya kak? Kak Jason: Sebentar ya, lagi ditanyakan</p> <p>Amriz: Tampilan program bakal mempengaruhi nilai apa engga ya kak? Kak Jason: Ngga, yang penting programnya berjalan dengan lancar seharusnya udah aman</p>
Tempat : Google Meet	
<p>Kehadiran Anggota Kelompok:</p> <p>No NIM Tanda tangan</p> <p>1</p>  <p>Ahmad Rizki 18221071</p> <p>2</p>  <p>Nadine Aliya Putri 18221081</p> <p>3</p>  <p>Clara Alrosa Fernanda Sinaga 18221099</p> <p>4</p>  <p>Pramaditva Fairi Migfar</p>	

<p>18221111</p>  <p>Naura Valda Prameswari 18221173</p>	
	<p>Tanda Tangan Asisten:</p>  <p>Jason Stanley Yoman 13519019</p>

9.3 Log Activity Anggota Kelompok

Nama	Tanggal	Aktivitas
Ahmad Rizki / 18221071	22 November 2022	Menambahkan checkpoint
Naura Valda P/ 18221173	22 November 2022	Memperbaiki dan menambahkan ADT
Naura Valda P/ 18221173	23 November 2022	Menambahkan rancangan dasar game Snake on Meteor
Naura Valda P/ 18221173	24 November 2022	Memperbahurui ADT stackGame Menambahkan fungsi moveSnake pada game

		Snake on Meteor
Nadine Aliya Putri / 18221081	24 November 2022	Menambahkan fungsi history dan reset history
Ahmad Rizki / 18221071	24 November 2022	Menambahkan ADT mapGame Memperbaharui game Snake on Meteor Memperbaharui ADT <i>linked list</i>
Clara Alrosa Fernanda Sinaga / 18221099	24 November 2022	Membuat file country.txt untuk keperluan game Hangman Membuat dasar game Hangman
Naura Valda P/ 18221173	25 November 2022	Memperbaiki history dan reset history Memperbaharui load Memperbaharui game Snake on Meteor
Ahmad Rizki / 18221071	25 November 2022	Membantu memperbaiki history
Clara Alrosa Fernanda Sinaga / 18221099	25 November 2022	Memperbaharui game Hangman
Naura Valda P/ 18221173	26 November 2022	Memperbaiki history Menyelesaikan scoreboard Memperbaiki load
Ahmad Rizki / 18221071	26 November 2022	Memperbaiki scanparser Memperbaharui ADT gameStack Menambahkan peta
Clara Alrosa Fernanda Sinaga / 18221099	27 November 2022	Menambahkan score pada game Hangman Memperbaiki game Hangman
Naura Valda P/ 18221173	27 November 2022	Memperbaiki save Memperbaharui scoreboard Menyelesaikan revisi Tugas Besar 1
Ahmad Rizki / 18221071	27 November 2022	Memperbaharui game Snake on Meteor Menambahkan score pada game Snake on Meteor

Nadine Aliya Putri / 18221081	27 November 2022	Memperbaharui history
Naura Valda P/ 18221173	28 November 2022	Memperbaharui game Snake on Meteor Memperbaiki scoreboard Menyelesaikan reset scoreboard Memperbaiki main program
Ahmad Rizki / 18221071	28 November 2022	Memperbaiki scoreboard Memperbaharui history
Pramaditya Fajri Migfar / 18221111	29 November 2022	Menambahkan ADT stackTOH Menyelesaikan dan menambahkan game Tower of Hanoi
Ahmad Rizki / 18221071	29 November 2022	Memperbaharui main program
Clara Alrosa Fernanda Sinaga / 18221099	29 November 2022	Memperbaharui game Hangman
Pramaditya Fajri Migfar / 18221111	30 November 2022	Memperbaharui game Tower of Hanoi
Ahmad Rizki / 18221071	30 November 2022	Memperbaiki game Hangman
Naura Valda P/ 18221173	30 November 2022	Menambahkan bonus game Treasure Hunt
Clara Alrosa Fernanda Sinaga/18221099 Nadine Aliya Putri/18221081	30 November 2022	Membuat laporan
Ahmad Rizki / 18221071	1 Desember 2022	Memperbaiki bug program
Naura Valda P/ 18221173	1 Desember 2022	Memperbaiki tampilan game Treasure Hunt

Semua Anggota	1 Desember 2022	Membuat laporan Asistensi Debugging
---------------	-----------------	---