

LAPORAN TUGAS BESAR

IF2111/Algoritma dan Struktur Data

Mobitangga


Dipersiapkan oleh:

Kelompok 4

18220047 Dhiya Risalah Ghaida
18220084 Dewa Ayu Mutiara Kirana P D
18220059 Faiza Aqiela Zuma
18220030 Shafira Putri Azmi Imani
18220032 Rifki Kaida
18219098 Muhamad Agung Ahsary

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		<i>IF2111-TB-04</i>		26
		<i>Revisi</i>	<i>1</i>	<i>28 November 2021</i>

Daftar Isi

Ringkasan	3
Penjelasan Tambahan Spesifikasi Tugas	4
Fitur Banner	4
Fitur Tambahan Loading	5
Struktur Data (ADT)	5
ADT Mesin Karakter dan Mesin Kata	6
ADT Array Dinamis	6
ADT List Linier	6
ADT Stack	7
ADT Map	7
ADT Player	7
ADT Teleporter	7
ADT Roll	8
ADT Skill	8
Program Utama	9
Algoritma-Algoritma Menarik	11
Pemakaian fungsi strcmp()	11
Pemakaian fungsi rand() dan srand()	12
Pemakaian fungsi fflush(stdout), usleep(), dan fputc()	13
Pemakaian ADT Player	15
Pembagian Kerja dalam Kelompok	17
Lampiran	18
Deskripsi Tugas Besar	18
Notulen Rapat	19
Log Activity Anggota Kelompok	23

1 Ringkasan

Mobitangga merupakan sebuah permainan yang konsepnya serupa dengan *board game* ular tangga, namun digitalisasi dan dimodifikasi. Terdapat berbagai fitur baru yang dapat membuat permainan ini terlihat lebih menarik. Contohnya, pemain dapat memiliki *skill-skill* yang beragam dengan efek yang berbeda-beda. Selain itu, peta pada permainan ini terdiri dari petak - petak yang digambarkan dengan tanda titik (".") sebagai petak kosong dan tanda pagar ("#") yang menunjukkan petak terlarang. Petak terlarang mengakibatkan pemain tidak dapat berada pada petak tersebut. Pada peta juga tersebar petak - petak teleportasi. Teleporter ini digambarkan dengan dua buah bilangan, yang masing - masing menunjukkan lokasi teleporter dan petak keluar teleporter.

Saat program dimulai, pemain akan disambut oleh *banner* permainan dan dihadapkan dengan pilihan main menu yang untuk memulai permainan atau keluar dari permainan. Sebelum memulai, *user* akan diminta untuk memberi masukkan data pemain, seperti jumlah orang yang akan bermain serta nama dari setiap pemain. Setelah itu, program akan membaca *file* konfigurasi yang telah dibuat dengan tujuan untuk memetakan arena bermain dari mulai panjang permainan, jumlah maksimal *roll*, hingga letak teleportasi. Setiap giliran, pemain akan diberikan satu *skill* secara acak dan disimpan pada *container skill* masing - masing pemain dengan jumlah maksimal *skill* sebanyak 10. Adapun macam - macam *skill* pada permainan yaitu Pintu Ga Ke Mana - Mana, Cermin Pengganda, Senter Pembesar Hoki, Senter Pengecil Hoki, Mesin Penukar Posisi, hingga Teknologi Gagal (pemain tidak mendapatkan *skill* apapun).

Program akan diberi *input* yang berupa *command* oleh pemain dan program akan menjalankan perintah sesuai dengan masukan dari pemain tersebut. Program ini dijalankan dengan menggunakan *command prompt* yang secara garis besar dapat melakukan perintah seperti SKILL, MAP, ROLL, INSPECT, ENDTURN, dan UNDO. Permainan akan berakhir apabila sudah terdapat pemain yang sampai ke petak N, yaitu panjang maksimal peta permainan yang disesuaikan berdasarkan *file* konfigurasi yang telah dibuat. Di akhir permainan, program akan menampilkan urutan peringkat permainan berdasarkan posisi dari setiap pemain.

Permainan Mobitangga diimplementasikan dengan memakai bahasa C berbasis *text* melalui *command line* (CLI) dan memanfaatkan ADT yang telah dipelajari pada mata kuliah IF2111 - Algoritma dan Struktur Data. Namun, terdapat pula beberapa ADT tambahan yang dibuat guna mendukung keberjalanan program ini agar lebih efektif dan efisien.

Laporan ini mencakup ringkasan permainan, penjelasan tambahan spesifikasi tugas, penjelasan struktur data yang dipakai, penjelasan program utama, algoritma-algoritma menarik, dan lampiran penting lainnya.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 Fitur Banner

Fitur ini sebenarnya hanya fitur kosmetik untuk memperindah tampilan sehingga benar-benar merepresentasikan permainan yang telah dirancang. Dapat dilihat pada gambar 2.1.1, terdapat judul dari permainan ini dan gambar ular yang merepresentasikan permainan ular tangga pada permainan papan. Perancangan *banner* ini menggunakan fitur *ascii art generator* untuk bagian tulisan dan gambarnya.

Gambar 2.1.1 *Opening banner game*



2.2 Fitur Tambahan Loading

Fitur ini merupakan fitur kosmetik tambahan untuk memperindah tampilan dari game yang telah dirancang. Bentuk implementasinya merupakan *output string* yang telah ditentukan sehingga seolah-olah akan seperti sedang *loading*.

Gambar 2.1.2 *Code* Fitur Tambahan *Loading*

```
void printRD(){
    const int trigger = 500; // ms
    const int numDots = 4;
    const char prompt[] = "Sedang membaca konfigurasi";

    for (int i= 0; i < 4; i++) {
        // Return and clear with spaces, then return and print prompt.
        printf("\r%s\r%s", sizeof(prompt) - 1 + numDots, "", prompt);
        fflush(stdout);

        // Print numDots number of dots, one every trigger milliseconds.
        for (int i = 0; i < numDots; i++) {
            usleep(trigger * 1000);
            fputc('.', stdout);
            fflush(stdout);
        }
    }
    printf("\n");
}
```

3 Struktur Data (ADT)

Struktur data yang digunakan pada permainan Mobitangga adalah ADT Mesin Karakter dan Mesin Kata, Array Dinamis, List Linier, Stack, Map, dan Player

3.1 ADT Mesin Karakter dan Mesin Kata

ADT Mesin Karakter dan Mesin Kata digunakan untuk membaca file konfigurasi yang berisi informasi peta, seperti panjang peta, jenis string petak (tanda titik (.) atau tanda pagar (#)), bilangan MaxRoll, jumlah *teleporter*, dan letak *teleporter* berada. Selain itu, ADT ini juga digunakan untuk membaca *command* dari pemain.

Fungsi dari ADT Mesin Karakter adalah untuk membaca suatu karakter pada pita dan kemudian menyimpan karakter tersebut sebagai variabel CC. Jika CC adalah '\n' atau MARK, maka EOP (*End of Process*) akan bernilai *TRUE* dan menandakan akhir dari pita. Mesin karakter dibutuhkan untuk pengimplementasian mesin kata. Sedangkan pada Mesin Kata, terdapat struktur data Kata yang terdiri oleh elemen TabKata sebagai *container* penyimpan kata dengan indeks dari 1 sampai NMax dan juga elemen Length dalam *integer*.

ADT Mesin Karakter memiliki primitif START yang berguna untuk membaca *file* konfigurasi yang berbentuk “.txt”. Selain itu terdapat primitif START_Command yang bertujuan untuk menyiapkan pita untuk dibaca. Terakhir, terdapat prosedur ADV yang bertujuan untuk membaca karakter selanjutnya pada pita.

Pada Mesin Kata, terdapat primitif yang serupa oleh Mesin Karakter yaitu STARTKATA, STARTforCommand, ADVKATA. Namun, terdapat pula prosedur tambahan seperti IgnoreBlank yang bertujuan untuk mengabaikan satu atau beberapa blank (‘ ’) dan prosedur SalinKata yang berfungsi untuk mengakuisisi kata dan menyimpannya dalam CKata.

ADT Mesin Karakter dikonfigurasi pada header “mesin_kar.h” dan diimplementasikan pada file “mesin_kar.c”, sedangkan ADT Mesin Kata dikonfigurasi pada *header* “mesin_kata.h” dan diimplementasikan pada file “mesin_kata.c”.

3.2 ADT Array Dinamis

ADT Array Dinamis digunakan untuk mempresentasikan tampilan peta dan *teleporter*. Pada ADT ini, terdapat struktur data TabInt yang berisi elemen Neff dalam bentuk *integer* yang menjelaskan banyak elemen efektif. Kemudian, terdapat elemen TI dalam bentuk ElType yang merupakan memori tempat menyimpan elemen (*container*). Tipe data TabInt dimanfaatkan untuk membuat tipe Map pada ADT Map. Konsep ADT ini juga dimanfaatkan untuk membuat tipe data pada ADT Player. Elemen TabBuff pada tipe DataPlayer dan TabPlayer pada tipe Player mengikuti konstruksi tipe data TabInt. ADT Array Dinamis dikonfigurasi pada *header* “array.h” dan diimplementasikan pada file “array.c”.

3.3 ADT List Linier

ADT List Linier merupakan struktur yang digunakan untuk menyimpan *skill* yang didapat oleh pemain pada awal ronde dan juga untuk membuang *skill* yang dimiliki oleh pemain. Pada List Linier, terdapat dua tipe data yang terdefinisi yaitu ElmtList yang terdiri dari elemen info dalam bentuk *infotype* dan elemen *Next* dalam bentuk *address*. Selain itu, terdapat pula tipe data List dengan elemen *First* dalam bentuk *address*.

Primitif yang digunakan merupakan primitif yang menunjang ADT List itu sendiri karena melalui primitif yang digunakan, kita dapat menggunakan selektor dan konstruktor untuk menampilkan informasi list *skill*, mencari dan mengeluarkan informasi *skill* tertentu, dan penambahan/pengurangan *skill* pada *list*. Dalam ADT ini, terdapat berbagai macam konstruktor, seperti IsEmptyList untuk mengecek suatu *list* itu kosong atau tidak, CreateEmpty untuk membuat sebuah *list* kosong, alokasi dan dealokasi *list*, insertFirst dan insertLast guna *insert*, serta DelFirst, DelLast, dan DeleteI guna menghapus elemen dari sebuah *list*. Selain itu, juga terdapat konstruktor untuk mengecek jumlah elemen menggunakan fungsi NbElmt serta PrintInfo dan infoKe untuk mengecek info dari suatu *list*. ADT List Linier dikonfigurasi pada *header* “listlinier.h” dan diimplementasikan pada file “listlinier.c”.

3.4 ADT Stack

ADT Stack digunakan untuk melakukan *command undo*. ADT Stack adalah list linear yang bekerja dengan mengenali elemen puncaknya yang ditunjukkan dengan sebuah pointer bernama TOP. Kemudian, elemen puncak tersebut merupakan elemen yang akan disisipkan elemen baru atau dihapus elemennya. Dengan kata lain, TOP adalah satu-satunya alamat tempat terjadi operasi. ADT Stack dikonfigurasi pada *file header* “stack.h” dan diimplementasikan pada *file* “stack.c”.

3.5 ADT Map

Pada ADT Map tersimpan tipe data Map yang menyimpan informasi panjang peta, *maxroll*, jumlah teleporter, konfigurasi peta, dan letak *teleporter* pada peta. Terdapat primitif prosedur *ConfigureMap* untuk membaca *file* konfigurasi menjadi peta dan fungsi *ConverterCharInt* untuk mengubah karakter menjadi *integer*. ADT Map dikonfigurasi pada *file header* “map.h” dan diimplementasikan pada *file* “map.c”.

3.6 ADT Player

Pada ADT Player terdapat 3 struktur yaitu *BuffPlayer*, *DataPlayer*, dan *Player*. *BuffPlayer* terdiri atas *isImmuned* (*boolean*) yang mengecek kondisi pemain *immune* terhadap *teleporter* atau tidak, *isSenterPembesar* (*boolean*) untuk mengecek pemain sudah mendapatkan *buff* dari *skill* Senter Pembesar Hoki atau tidak, *isSenterPengecil* (*boolean*) untuk mengecek pemain telah sudah mendapatkan *buff* dari *skill* Senter Pengecil Hoki atau tidak, dan *isCerminGanda* (*boolean*) untuk menentukan pemain telah mendapatkan *buff* dari *CerminPengganda* atau tidak. Struktur-struktur ini digunakan untuk mengetahui kondisi pemain apabila menggunakan *skill* yang dimiliki.

Kemudian, *DataPlayer* merupakan struktur data yang menyimpan informasi yang dimiliki oleh seorang pemain yang terdiri atas nama pemain (*string*) guna merepresentasikan nama dari pemain, *nomorPemain*(*integer*) guna merepresentasikan nomor dari pemain, *position* (*integer*) gunamenunjukkan posisi pemain pada peta, *skills* (*list*) yang merupakan daftar dari *skill* yang dimiliki oleh pemain, *isRolled* (*boolean*) untuk menunjukan pemain sudah melakukan roll pada turn tersebut atau belum, *isTeleported* (*boolean*) yang menunjukan pemain sedang mendarat pada petak teleporter atau tidak, dan *TabBuff*(*BuffPlayer*) yang menyimpan kondisi-kondisi *buff* pada pemain. Lalu, terdapat struktur data *Player* yang terdiri atas *jumlahPlayer*(*integer*) yang merepresentasikan jumlah pemain yang bermain dan *TabPlayer*(*DataPlayer*) yang merupakan struktur data yang menghimpun data dari pemain. ADT Player dikonfigurasi pada *file header* “player.h” dan diimplementasikan pada *file* “player.c”.

3.7 ADT Teleporter

ADT Teleporter memiliki satu fungsi primitif yaitu *teleport* yang berguna untuk memindahkan pemain ke petak lain saat menginjak petak yang memiliki teleporter. ADT Teleporter menggunakan ADT map untuk mengkonfigurasi peta dan mengetahui letak-letak

teleporter. ADT Teleporter dikonfigurasi pada *file header* “teleporter.h” dan diimplementasikan pada file “teleporter.c”.

3.8 ADT Roll

ADT Roll memiliki dua primitif yaitu *randomize* dan *roll* yang berguna untuk memindahkan pemain dari satu petak ke petak lain dengan syarat pemain harus melempar dadu untuk mendapatkan nilai dadu. Kemudian, pemain dapat maju ataupun mundur dengan syarat petak yang dituju *valid* (di antara 1 dan N) dan bukan petak terlarang. ADT Roll menggunakan ADT Map untuk mengkonfigurasi peta dan mengetahui letak-letak petak yang valid untuk dituju, ADT Time untuk mendapatkan nilai dadu yang *random*. Selain itu ADT Roll juga menggunakan ADT Mesin Kata dan ADT Array. ADT Roll dikonfigurasi pada file header “roll.h” dan diimplementasikan pada file “roll.c”.

3.9 ADT Skill

ADT Skill di konfigurasi pada file *header* skill.c. Dalam aplikasinya, ADT Skill menggunakan bantuan ADT List Linear yang berguna untuk menyimpan *skill* yang didapat oleh pemain pada awal ronde dan juga untuk membuang *skill* yang dimiliki oleh pemain. ADT Skill memiliki primitif *chance* yang dapat digunakan untuk menentukan persentase mendapatkan *skill* tiap *turn* yang nantinya akan diimplementasikan melalui prosedur *getSkill* dan *getSkill2*. Penjelasan mengenai kedua prosedur tersebut akan dijelaskan lebih lanjut pada bagian algoritma menarik terkait alasan kedua fungsi tersebut harus dipisah. Selanjutnya, terdapat prosedur *getSkill* yang berfungsi untuk melakukan randomisasi untuk pemain mendapatkan *skill* tiap turnnya. Setiap *skill* yang didapat akan dialokasikan ke sebuah *list* dengan batasan maksimal sebanyak sepuluh *skill*. Apabila sudah melebihi batas tersebut, maka program akan mengeluarkan *output* yang memberikan informasi bahwa jumlah *skill* pada pemain tersebut sudah penuh. Pemain juga memungkinkan untuk tidak mendapatkan *skill* apabila saat melakukan randomisasi pemain malah mendapatkan *skill* Teknologi Gagal.

Selanjutnya, terdapat prosedur *displaySkill* yang dapat menampilkan *skill* yang telah didapatkan oleh pemain. Pertama, didefinisikan terlebih dahulu terutama pada bagian *skill* yang bisa didapatkan pada bagian *char* *listSkill*. Setelah itu, *main program* akan mengimplementasikan dan menampilkan *list skill* pemain. Selanjutnya, terdapat prosedur *BuangSkill* untuk menghapus *skill* yang dimiliki pemain. Seperti halnya prosedur *displaySkill*, didefinisikan terlebih dahulu *skill* apa saja yang akan didapatkan oleh pemain. Kemudian, dilakukan pengecekan pemain memiliki *skill* atau tidak. Jika pemain memiliki *skill*, maka *skill* akan dihapus sesuai dengan *input* dari *user*. Jika pemain tidak memiliki *skill*, maka akan ditampilkan *output* pemberitahuan yang menyatakan pemain tidak memiliki *skill*. Selanjutnya, terdapat prosedur *cerminPengganda* yang merupakan implementasi dari *skill* Cermin Pengganda. Prosedur ini menggunakan prosedur *getSkill2* sebanyak dua kali untuk mendapatkan dua *skill*. Terakhir, terdapat prosedur *skillSebelum* yang akan diimplementasikan untuk proses *undo* pada *main program*.

4 Program Utama

Program utama pada permainan mobitangga berisi *code-code* utama keberjalanan permainan yang mengintegrasikan seluruh fungsi dari gabungan ADT yang telah dibuat. Pada awalnya, main program akan memanggil fungsi *ConfigurasiMap* dan mulai melakukan konfigurasi peta dan membuat *player*. Proses membuat *player* diawali dengan *looping* pengecekan *input* jumlah pemain yang valid dalam permainan ($2 \leq \text{jumlah pemain} \leq 4$). Jika *input* tidak valid, maka program akan meminta *user* untuk *input* ulang (asumsi : nama pemain tanpa spasi).

Program akan menerima jumlah yang bermain dan menyimpan pada *address Player* (array yang menyimpan jumlah *player*). Terdapat inisialisasi urutan giliran permainan $n = 1$ dan program akan melakukan *looping* giliran hingga jumlah pemain hingga ditemukan pemenang dalam permainan (posisi pemain = panjang map). Dalam keberjalanan permainan, untuk setiap *turn* (ronde), pemain yang belum melakukan *roll* dadu akan diinisialisasi dengan kondisi FALSE.

Jika proses konfigurasi dan pembuatan *player* telah selesai, program akan meminta *input*-an *user* untuk menggunakan *command* yang valid. *Command-command* yang akan digunakan dalam permainan berada pada *file* terpisah, namun tentu terintegrasi dengan program utama. Pada *file* *command.c*, terdapat beberapa fungsi yang digunakan untuk mengecek *input*-an *user*, yang diproses melalui mesin kata. Sebagai contoh, jika *user* men-*input* "SKILL/skill", fungsi *isSkill* akan mengeluarkan TRUE. Hal yang sama juga berlaku pada fungsi *isMap*, *isBuff*, *isInspect*, *isRoll*, *isSave*, *isUndo* dan *isEndturn*. Parameter yang digunakan pada fungsi-fungsi tersebut adalah *map*, *player*, dan giliran.

Pada fungsi *isSkill*, jika kondisi TRUE (*input* : SKILL/skill), maka program akan mengecek jumlah *skill* yang dimiliki pemain. Jika jumlah *skill* = 0, maka program akan menampilkan pesan "Kamu ga punya *skill* apa-apa nih!". Namun, jika jumlah *skill* != 0, maka program akan menampilkan daftar *skill* yang dimiliki oleh pemain. Selain itu, program memberikan pilihan pada *user* untuk memberikan perintah pada program, yaitu "tekan 0" untuk keluar dari menu *skill* dan "bilangan negatif" jika ingin membuang *skill*. Jika pengguna ingin menggunakan *skill* tertentu, maka pengguna harus memasukkan angka sesuai dengan nomor *skill* yang terdapat pada daftar *skill*, bersesuaian dengan nama *skill* yang ingin digunakan oleh pemain. *Skill* yang telah digunakan akan terhapus dari daftar *skill*. Perintah masukan angka harus selalu diisi dengan tipe data integer. Jika *input user* tidak sesuai, program akan meminta *input user* kembali.

Jika fungsi *isMap* berada pada kondisi TRUE (*input* : MAP/map), maka program akan mengeluarkan peta seluruh pemain berdasarkan *file* konfigurasi peta dan posisi pemain. Peta pada permainan ini terdiri dari petak-petak yang digambarkan dengan tanda titik (.). Terdapat petak-petak pengecualian, yaitu petak yang ditandai tanda pagar (#) yang menunjukkan petak terlarang. Penanda posisi pemain pada peta ditandai dengan tanda bintang (*). Pada fungsi *isBuff*, fungsi ini berkaitan erat dengan pemakaian *skill* pada *player*. Jika *player* sedang menggunakan *skill*, maka ketika perintah ini dimasukkan (*isBuff* TRUE, *input* = BUFF/buff), program akan menampilkan daftar *buff* yang sedang aktif. Sebaliknya, jika *buff* = 0 (kondisi inisialisasi), maka program akan menampilkan pesan "Belum ada *buff* yang aktif".

Pada kondisi `isInspect TRUE` (`input = INSPECT/inspect`), user akan diminta untuk memasukkan nomor petak yang berupa integer. Setelah itu, program akan mengecek `input`-an `user` berdasarkan `file` konfigurasi. Apabila nomor yang dimasukan bertepatan dengan nomor petak terlarang (#), maka program akan memberikan informasi petak tersebut merupakan petak terlarang. Di luar kondisi tersebut, jika `input`-an user bertepatan dengan petak `teleporter`, program juga akan memberikan informasi petak `teleporter` berdasarkan `file` konfigurasi. Informasi yang ditampilkan menunjukkan lokasi `teleporter` dan petak keluar `teleporter`. Jika `input`-an user tidak memenuhi kedua kondisi tersebut (petak terlarang dan `teleporter`), maka program akan menampilkan pesan “Petak merupakan petak kosong”.

Apabila `isRoll TRUE` (`input = ROLL/roll`), program akan kembali mengecek kondisi `player`. Jika `player` sudah melakukan `roll`, maka program akan menampilkan pesan “Tidak bisa `roll` dua kali”. Sebaliknya, jika `player` belum pernah melakukan `roll`, maka program akan mengecek kondisi `skill` yang sedang digunakan oleh pemain. Jika pemain sedang menggunakan senter pembesar, maka hasil `roll` dadu akan berada pada range 5 - 10. Jika `skill` yang digunakan adalah senter pengecil, maka hasil `roll` dadu berada pada range 1 - 5. Jika bukan keduanya, maka hasil `roll` dadu merupakan angka random dari 1-10 (10 merupakan `MaxRoll`).

Apabila `player` sudah melakukan `roll` dadu, program akan melakukan perhitungan posisi `user` setelah ditambah/dikurang dari hasil dadu. Berdasarkan hasil perhitungan tersebut, program akan menampilkan pilihan pada user untuk memilih pergerakan pemain, baik maju atau mundur. Pilihan user tersebut akan diintegrasikan dengan posisi `player` pada `map` pemain tersebut. Jika user sudah memilih, program akan kembali melakukan pengecekan. Jika petak hasil perhitungan tidak berada pada petak terlarang (#) dan valid pada peta, maka `player` akan bergerak menuju petak tersebut. Sebaliknya, jika petak hasil perhitungan tidak valid pada `map` atau menunjukkan peta terlarang, program akan menampilkan pesan “Tidak dapat bergerak”. Apabila giliran suatu pemain sudah selesai, `user` dapat memasukkan input “ENDTURN/endturn. Pada kondisi pemain sudah melakukan `roll` dadu, fungsi ini akan bernilai `TRUE`. Sebaliknya, jika pemain belum melakukan `roll` dadu, program akan menampilkan pesan “Harus `roll` dadu terlebih dahulu, baru bisa `endturn`”.

Permainan berakhir ditandai dengan ditemukannya posisi pemain yang sudah sama dengan panjang peta. Pemain yang sudah mencapai posisi tersebut merupakan pemenang dalam permainan Mobitangga ini. Setelah permainan berakhir, program akan menampilkan urutan peringkat pemain berdasarkan lokasi pemain.

5 Algoritma-Algoritma Menarik

5.1 Pemakaian fungsi strcmp()

Pada program permainan Mobitangga, fungsi strcmp() digunakan dalam ADT Player. Fungsi ini merupakan salah satu fungsi yang berada dalam *library* string.h. *Library* ini merupakan *library* yang menyimpan fungsi-fungsi untuk menangani/manipulasi string ataupun substring. Fungsi strcmp() ini sendiri berfungsi untuk membandingkan dua buah string.

Gambar 5.1.1 Header file pada ADT Player

```
#include "player.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "boolean.h"
#include "listlinier.h"
#include "skill.h"
```

Gambar 5.1.2 Implementasi fungsi strcmp()

```
for (j = 1; j <= banyakPlayer; j++) {
    if ((!strcmp((*P).TabPlayer[i].pName, (*P).TabPlayer[j].pName)) && (i != j)){
        salah++;
        printf("Nama %s sudah ada!\n", (*P).TabPlayer[i].pName);
        printf("Ganti nama player-%d: ", i);
        scanf("%s", ((*P).TabPlayer[i].pName));
    }
}
```

Fungsi ini dianggap menarik karena menunjukkan operasi/manipulasi *string* yang membandingkan dua *input* nama player. Algoritma pada program memperlihatkan proses *looping* yang mengecek *input*-an user. Bila ditemukan kesamaan nama yang dimasukkan (string memiliki value yang sama), maka program akan meminta user untuk melakukan input ulang.

5.2 Pemakaian fungsi rand() dan srand()

Pada dasarnya, fungsi rand() merupakan fungsi yang dapat melakukan *generate* angka secara acak. Namun, jika dilakukan secara berurutan, angka yang dihasilkan akan sama setiap kali program dijalankan. Oleh karena itu, diperlukan fungsi srand() guna mengatur titik awal untuk menghasilkan sekumpulan *pseudo-random integers*. Srand() tidak mengembalikan nilai apapun, hanya menginisialisasi bilangan acak semu yang dihasilkan oleh fungsi rand(). Fungsi - fungsi ini dapat digunakan ketika program telah men-include library stdlib.h.

Gambar 5.2.1 Header file pada ADT Skill

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "map.h"
#include "mesin_kata.h"
#include "mesin_kar.h"
#include "array.h"
```

Fungsi rand() dan srand() pada program Mobitangga digunakan pada ADT Skill dan ADT Roll. Pada ADT Roll, fungsi ini digunakan untuk men-*generate* angka dadu yang akan diperoleh oleh *player* saat memainkan permainan. Angka yang dihasilkan memiliki batasan angka maksimal yang disesuaikan dengan *file* konfigurasi. Dalam implementasinya, fungsi rand() dan srand() didukung oleh fungsi time_t sebagai benih pengacakan yang "tidak dapat diprediksi", kecuali jika program dijalankan pada waktu yang sama. Fungsi time_t ini diperoleh melalui library time.h yang berfungsi untuk memanipulasi waktu. Pada ADT Skill, fungsi ini digunakan untuk men-*generate skill-skill* yang akan diperoleh player berdasarkan persentase *chance* yang ada. Pada program Mobitangga, terdapat dua fungsi untuk men-*generate* skill, yaitu getSkill dan getSkill2.

Gambar 5.2.2 Implementasi rand() dan srand() pada ADT Roll

```
int randomize(time_t t, int Max){
    return rand() % Max;
}

int roll(int buffSkill, int maxRoll){
    time_t t;
    srand((unsigned) (time(&t)));
```

Gambar 5.2.3 Implementasi rand() dan srand() pada fungsi getSkill

```
void getSkill(List *skill){
    int x;
    time_t t;

    srand(time(0));
    x = rand();
    x = chance((x) % 100 + 1);
    if (x == 1){
        printf("Pintu Ga Ke Mana-Mana\n");
    }
}
```

Gambar 5.2.4 Implementasi rand() dan srand() pada fungsi getSkill2

```
void getSkill2(List *skill){
    int x;

    x = rand();
    x = chance((x) % 100 + 1);
}
```

Algoritma - algoritma ini dianggap menarik karena dengan adanya keterlibatan `srand()` dan `time_t`, ternyata dapat memberikan perbedaan hasil yang sangat signifikan bila dibandingkan dengan fungsi yang hanya menggunakan fungsi `rand()` saja. Hal ini terlihat pada implementasi fungsi `getSkill` dan `getSkill2` pada ADT Skill. Pada fungsi `getSkill`, skill hasil proses *generate* diharapkan selalu acak setiap waktunya sehingga diperlukan fungsi `time_t` untuk memanipulasi waktu. Sebaliknya, fungsi `getSkill2` digunakan sebagai implementasi dari *skill* cerminPengganda yang memerlukan *output* acak yang memiliki *value* yang sama. Oleh karena itu, pada `getSkill2` fungsi yang digunakan hanya fungsi `rand()` saja.

5.3 Pemakaian fungsi `fflush(stdout)`, `usleep()`, dan `fputc()`

Pada umumnya, fungsi `fflush()` digunakan hanya untuk output stream guna membersihkan (atau *flush*) *output buffer* dan memindahkan data *buffer* menuju *console*. Hal ini untuk memastikan bahwa apapun yang baru saja ditulis ke dalam file/console memang ditulis di dalam konsol. Selain itu, fungsi `usleep()` memiliki fungsi untuk menghentikan/menunda program berjalan dalam satuan *microseconds*. Fungsi ini dapat digunakan jika program telah *men-include library* `unistd.h`. Berbeda dengan kedua fungsi tersebut, `fputc` merupakan salah satu operasi file yang berfungsi untuk menyimpan data bertipe string ke dalam file.

Gambar 5.3.1 Implementasi fflush(stdout), usleep(), dan fputc() (a)

```
void printLoading(){
    const int trigger = 500; // ms
    const int numDots = 4;
    const char prompt[] = "Loading";

    for (int i= 0; i < 2; i++) {
        // Return and clear with spaces, then return and print prompt.
        printf("\r%s\r%s", sizeof(prompt) - 1 + numDots, "", prompt);
        fflush(stdout);

        // Print numDots number of dots, one every trigger milliseconds.
        for (int i = 0; i < numDots; i++) {
            usleep(trigger * 1000);
            fputc('.', stdout);
            fflush(stdout);
        }
    }
    printf("\n");
}
```

Gambar 5.3.2. Implementasi fflush(stdout), usleep(), dan fputc() (b)

```
void printRD(){
    const int trigger = 500; // ms
    const int numDots = 4;
    const char prompt[] = "Sedang membaca konfigurasi";

    for (int i= 0; i < 4; i++) {
        // Return and clear with spaces, then return and print prompt.
        printf("\r%s\r%s", sizeof(prompt) - 1 + numDots, "", prompt);
        fflush(stdout);

        // Print numDots number of dots, one every trigger milliseconds.
        for (int i = 0; i < numDots; i++) {
            usleep(trigger * 1000);
            fputc('.', stdout);
            fflush(stdout);
        }
    }
    printf("\n");
}
```

Gambar 5.3.3 Implementasi fflush(stdout), usleep(), dan fputc() (3)

```
void printRoll(){
    const int trigger = 500; // ms
    const int numDots = 4;
    const char prompt[] = "Rolling";

    for (int i= 0; i < 2; i++) {
        // Return and clear with spaces, then return and print prompt.
        printf("\r%s\r%s", sizeof(prompt) - 1 + numDots, "", prompt);
        fflush(stdout);

        // Print numDots number of dots, one every trigger milliseconds.
        for (int i = 0; i < numDots; i++) {
            usleep(trigger * 1000);
            fputc('.', stdout);
            fflush(stdout);
        }
    }
    printf("\n");
}
```

Pada permainan Mobitangga ini, ketiga fungsi tersebut diimplementasikan dalam ADT Console, yang merupakan percabangan dari program utama. Ketiga fungsi tersebut berada dalam fungsi printLoading, printRD, dan printRoll yang merupakan fungsi tambahan Algoritma ini dianggap menarik karena fungsi-fungsi tersebut digunakan untuk mengimplementasikan fitur kosemetik tambahan fitur kosmetik tambahan untuk memperindah tampilan permainan.

5.4 Pemakaian ADT Player

Gambar 5.4.1 *Header file* pada ADT Player

```
1 #ifndef __PLAYER_H__
2 #define __PLAYER_H__
3
4
5 #include <stdlib.h>
6 #include "boolean.h"
7 #include <stdio.h>
8 #include "listlinier.h"
9 #include "skill.h"
10
11 #define NilPlayer NULL
```



```

13 typedef struct{
14     boolean isImuned;
15     boolean isSenterPembesar;
16     boolean isSenterPengecil;
17     boolean isCerminGanda;
18 } BuffPlayer;
19
20 typedef struct{
21     char pName[15];
22     int nomorPemain;
23     int position;
24     List skills;
25     boolean isRolled;
26     boolean isTeleported;
27     BuffPlayer TabBuff[4+1];
28 } DataPlayer;
29
30 typedef struct{
31     int banyakPlayer;
32     DataPlayer TabPlayer[4+1];
33 } Player;
34
35 #define jumlah(Player) (Player).banyakPlayer
36
37 /***** KONSTRUKTOR *****/
38
39 void CreatePlayerEmpty(Player *P);
40
41 void InputPlayer(Player *P, int banyakPlayer);
42
43 void showName (Player *P, int nomorPemain);
44
45 void useSkill(Player *P, int inputs, int nomorPemain);
46
47 void skillSebelum(List aslinya, List *hasilcopy);
48
49 void showPeringkat(Player p);
50
51 #endif

```

ADT Player dibuat guna mengintegrasikan keseluruhan fungsi dalam setiap *Abstract Data Type* (ADT) yang telah dibuat. Hal ini dikarenakan “pemain” merupakan kunci utama dalam keberjalanan permainan. ADT ini menyimpan jumlah pemain yang bermain, skill yang dimiliki, kondisi buff, hingga posisi pemain dalam peta permainan.

Algoritma-algoritma dalam ADT Player dianggap menarik karena hanya dengan satu ADT, *user* dapat mengakses/mengintegrasikan keseluruhan ADT yang terdapat dalam program Permainan Mobitangga. Hal ini tentu akan membuat algoritma program jauh lebih efektif dan efisien.

6 Pembagian Kerja dalam Kelompok

Tabel 6.1 Pembagian Kerja dalam Kelompok

No.	Fitur/ADT	NIM Coder	NIM Tester
1	Program Utama (file main, console, banner)	18220084, 18220059	18220030
2	ADT Array	18220084, 18220047	18220030
3	Command	18220084	18220059, 18220032
4	ADT List Linier	18220059	18220030, 18220084
5	ADT Map	18220084, 18220032	18220059
6	ADT Mesin Karakter	18220084	18220059
7	ADT Mesin Kata	18220084	18220032, 18220047
8	ADT Player	18220047, 18220059, 18220084	18220030, 18220032
9	Roll	18220084	18220059
10	Skill	18220059	18220084
11	ADT Stack	18220047, 18220084	18220030
12	Teleporter	18220030	18220084

7 Lampiran

7.1 Deskripsi Tugas Besar

Sebuah Institut Teknologi tertentu sedang mengadakan lomba *game dev* dengan tema membuat board game digital terbaik se-kecamatan. Mendengar hal tersebut, Borakemon dan Mobita bersekongkol untuk membuat sebuah board game digital terasyik. Mereka kepikiran untuk menggabungkan game ular tangga dengan modifikasi-modifikasi yang dapat mengganggu lawan sehingga lahirlah ide Mobitangga.

2p

Namun sayangnya, Mobita tidak memiliki kemampuan maupun niat memprogram Mobitangga. Borakemon, kucing robot Mobita, juga belum memiliki kemampuan untuk memprogram karena belum belajar terlalu *deep*. Oleh karena itu, Borakemon menculik sekelompok *programmer* dari dimensi lain agar dapat membantu mereka membuat program Mobitangga agar dapat memenangkan lomba *game dev* itu.

Buatlah sebuah permainan berbasis CLI (*command-line interface*). Permainan 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.

7.2 Notulen Rapat

Asistensi I

Form Asistensi Tugas Besar


IF2110/Algoritma dan Struktur Data

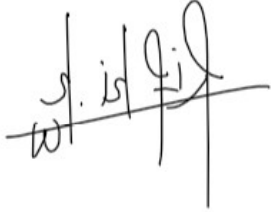

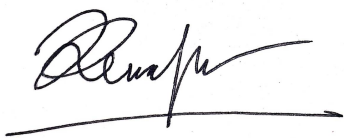

Sem. 1 2021/2022

No. Kelompok/Kelas : 04
Nama Kelompok :
Anggota Kelompok (Nama/NIM) :
1. 18219098 - Muhamad Agung Ahsary
2. 18220030 - Shafira Putri Azmi Imani
3. 18220032 - Rifki Kaida
4. 18220047 - Dhiya Risalah Ghaida
5. 18220059 - Faiza Aqiela Zuma
6. 18220084 - Dewa Ayu Mutiara Kirana P D

Asisten Pembimbing : 13518044 - Jun Ho Choi Hedyatmo

Asistensi I

Tanggal : 6 November 2021	Catatan Asistensi: Pertanyaan 1: Untuk map, posisi “#”, teleporternya fixed atau setiap mulai game berubah lagi Jawaban 1 : Itu dari file konfigurasi dan file konfigurasi itu kalian sendiri yang bikin. Based on hasil konfigurasi yang udah dibuat. Pertanyaan 2 : Chance di skill buat apa? Jawaban 2 : Nentuin peluang buat dapetin → Buat user yang main. Untuk nentuin kelangkaan skillnya. Pertanyaan 3 : Ada “.” dan “#” , itu bagaimana penjelasannya? Jawaban 3 :
Tempat : Google Meet	
Kehadiran Anggota Kelompok: No NIM Tanda tangan 1. 18219098 - Muhamad Agung Ahsary 2. 18220030 - Shafira Putri Azmi Imani  3. 18220032 - Rifki Kaida	

 <p>4. 18220047 - Dhiya Risalah Ghaida</p> <p>5. 18220059 - Faiza Aqiela Zuma</p>  <p>FAIZA A. ZUMA</p> <p>6. 18220084 - Dewa Ayu Mutiara Kirana P D</p>  <p>Dewa Ayu Mutiara</p>	<p>Itu kaya pagar gitu, kalo misal roll dadu, itu gabisa dilewatin. Kalau dapat # berarti gak bisa bergerak.</p> <p>Pertanyaan 4 : Kalo pemainnya di misal 96, trus dapet 6 saat roll dadu itu menang atau ga jalan atau mundur</p> <p>Jawaban 4 : Roll dadu bisa maju dan mundur, bisa maju atau mundur selama tidak ada “#” . Kalau gak memungkinkan keduanya, berarti tetap diam saja.</p> <p>Pertanyaan 5 : Yang berkaitan dengan ADT, itu harus sesuai atau kita bisa pakai ADT lain?</p> <p>Jawaban 5 : Kalo yang diwajibkan pake patokannya itu, tapi kalo ada detail implementasi yang lain itu terserah kalian.</p> <p>Pertanyaan 6 : Untuk buff itu gimana? Bukan fungsi kan?</p> <p>Jawaban 6 : Buff itu aktif dari skill dan bukan fitur gitu</p> <p>Pertanyaan 7 : Perihal driver, itu bagaimana?</p> <p>Jawaban 7 : Driver itu buat adtnya doang, jadi menguji fungsionalitas adtnya doang. Gak ada hubungannya sama programnya. Jadi skill game map itu gausah di driverin. Setiap ADT ada drivernya.</p>
	<p>Tanda Tangan Asisten:</p> 


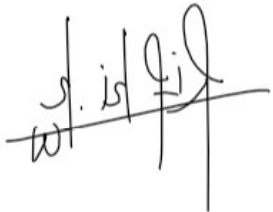
Asistensi II


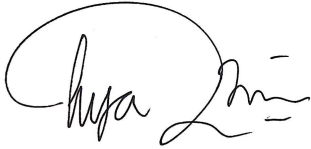
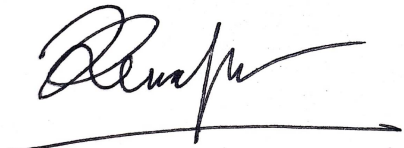
Form Asistensi Tugas Besar IF2110/Algoritma dan Struktur Data Sem. 1 2021/2022


No. Kelompok/Kelas : 04
Nama Kelompok :
Anggota Kelompok (Nama/NIM) : 1. 18219098 - Muhamad Agung Ahsary
2. 18220030 - Shafira Putri Azmi Imani
3. 18220032 - Rifki Kaida
4. 18220047 - Dhiya Risalah Ghaida
5. 18220059 - Faiza Aqiela Zuma
6. 18220084 - Dewa Ayu Mutiara Kirana P D

Asisten Pembimbing : 13518044 - Jun Ho Choi Hedyatmo

Asistensi II

Tanggal : 21 November 2021	Catatan Asistensi:
Tempat : Google Meet	
Kehadiran Anggota Kelompok: No NIM Tanda tangan 1. 18220030 - Shafira Putri Azmi Imani  2. 18220032 - Rifki Kaida 	
<p>Buff gabisa distack itu gimana? gaada buff yang aktif gitu kalo buffnya ga distack</p> <p>Untuk giliran main perlu buat adt apa ngga? gaperlu dibuat adt player, dibuat cycle aja. nanti tinggal dipakein mod buat aksesnya mau player ke berapa</p> <p>Ada saran untuk main gamenya nanti ga? kalo pengen belajar itu kan ada gameflow feature branch, nanti pull request buat gabunginnya. kalo gabungin emg gaada cara sendiri, kalian harus belajar cara make git, bikin branch per fitur, kalian kalo mau buat merge, merge per fiturnya. update histori kalo ada updatean baru. pokoknya ada game master yang main, nanti biar kalo ada updatean bisa di merge.</p> <p>Kalo undo trus ngehapus skill berarti kalo sebelumnya dapet cermin pengganda trus dipake, ngehapusnya langsung 2 skill + ada saran ga undo itu kan ngebalikin turn. gampangnya kalian ngecapture tiap turn kaya gimana. kalo ngecapture</p>	

<p>3. 18220059 - Faiza Aqiela Zuma</p>  <p>FAIZA A. ZUMA</p> <p>4. 18220047 - Dhiya Risalah Ghaida</p>  <p>5. 18220084 - Dewa Ayu Mutiara Kirana P D</p>  <p>Dewa Ayu Mutiara</p>	<p>itu semua buff, skill, dan info info lain itu disimpan, tinggal ambil 2 state di belakangnya gitu pake stuck</p> <p>kalo setelah ngeroll kan bisa maju mundur itu pure utak atik command roll kan yah? gaperlu di main programmnya lagi? terserah sih mau naruh dimana soalnya itu ngga terlalu penting juga.</p> <p>kalo misal playernya array tapi skillnya list sebenarnya bisa dikonekin ga sih? bisa bisa aja, tapi lebih baik keduanya disamain. kalian tinggal ngestore addressnya, nanti pakein pointer buat aksesnya</p> <p>Buat adt save itu dari tubes sebelumnya make matrix, jadi itu bagusnya gimana? tergantung gamenya, kalo contoh yang sekarang itu savenya bisa nyimpen kondisi pemain dan kondisi peta tiap saat, mereka di posisi mana, buff yang aktif itu apa aja. kita list semuanya dalam save file</p> <p>Berarti save mirip sama undo? iyaa bisa dibilang gitu, 1 adt turn bisa dipake sekalian buat save, load, sama undo.</p> <p>Laporannya itu gimana, pake notal kah? Penjelasan” itu diisi apa? gaperlu dibuat notalnya. tapi jelasin aja pake gambar or something, cara kerja adtnya ngapain, konsepnya gimana, dan bagaimana dia bisa membantu kerjanya. kalo yang algo menarik itu boleh aja dijelasin pake notal. penjelasan sketsa struktur data itu gaperlu pake kode, bisa pake kata kata/gambar aja.</p> <p>Algoritma menarik itu maksudnya gimana? ini bebas, contohnya kalian buat fungsi untuk ngerandom, kalo misal kalian mikir menarik itu boleh ditambahin. tp kalo gaada yg menarik gapapa klo ga ditambahin.</p> <p>Di adt player itu dia bakal ngecek dia di teleporter/engga dan punya imunitas/ engga. di spesifikasi dibilang kalo udah nginjek teleporter dia gabisa kena teleporter itu gimana? itu tentang buffnya, klo buffnya udah kepake sekali, jadi dia ga imun. jadi dia bisa kena teleport lagi</p> <p>Kita perlu bikin demo?</p>
---	---

	<p>kalo di spek ga ditulis, gausah</p> <p>Buff nya itu disimpen dimana? untuk buff yang aktif itu langsung disimpen di player aja.</p>
	<p>Tanda Tangan Asisten:</p> 

7.3 Log Activity Anggota Kelompok

Tabel 7.3.1 Log Activity Anggota Kelompok

No	Tanggal	NIM	Nama	Aktivitas
1	02/11/2021	18220032	Rifki Kaida	Membuat repository github
2	12/11/2021	18220032	Rifki Kaida	Mulai membuat ADT MAP
3	14/11/2021	18220084	Dewa Ayu Mutiara K P D	Memperbaiki ADT MAP, membuat dan memodifikasi ADT array, ADT mesin karakter dan ADT mesin kata dari praktikum.
4	15/11/2021	18220084	Dewa Ayu Mutiara K P D	Menyelesaikan ADT MAP dan push ke github
5	16/11/2021	18220084	Dewa Ayu Mutiara K P D	Membuat fungsi roll
6	16/11/2021	18220047	Dhiya Risalah Ghaida	Mulai membuat <i>code</i> program ADT Player
7	17/11/2021	18220059	Faiza Aqiela Zuma	Update ADT skill header, update beberapa fungsi pada ADT Skill serta debug implementasi skill
8	18/11/2021	18220059	Faiza Aqiela Zuma	Update ADT Skill skill
9	18/11/2021	18220084	Dewa Ayu Mutiara K P D	Mengupdate fungsi roll dan menyelesaikan fungsi roll tersebut, lalu dipush ke github
10	19/11/2021	18220030	Shafira Putri Azmi Imani	Membuat ADT Teleporter dan push ke github.

11	19/11/2021	18220047	Dhiya Risalah Ghaida	Menyelesaikan ADT Player lalu push ke github
12	19/11/2021	18220047	Dhiya Risalah Ghaida	Update ADT Player
13	19/11/2021	18220059	Faiza Aqiela Zuma	Update ADT List dan ADT Skill
14	20/11/2021	18220030	Shafira Putri Azmi Imani	Update Teleporter
15	20/11/2021	18220047	Dhiya Risalah Ghaida	Mulai membuat <i>code</i> program ADT State
16	20/11/2021	18220047	Dhiya Risalah Ghaida	Membuat ADT Player menggunakan array
17	20/11/2021	18220059	Faiza Aqiela Zuma	Mengkalibrasi ADT Skill dengan ADT Player, update Skill senter pengecil dan pembesar, melakukan set up dan mengkoneksi ADT Skill dan ADT Player
18	21/11/2021	18220047	Dhiya Risalah Ghaida	Menyelesaikan <i>code</i> program ADT State
19	21/11/2021	18220047	Dhiya Risalah Ghaida	Membuat command undo dan push ke github
20	21/11/2021	18220059	Faiza Aqiela Zuma	Mengetes skill dan player dengan memakai case array
21	22/11/2021	18220084	Dewa Ayu Mutiara K P D	Merevisi ADT player dan menyelesaikan ADT tersebut lalu dipush ke github. Membuat command inspect dan map lalu dipush ke github.
22	22/11/2021	18220059	Faiza Aqiela Zuma	Update ADT Skill dan membuat fungsi membuang skill. Serta update untuk main skill dan fungsi use skill
23	22/11/2021	18220030	Shafira Putri Azmi Imani	Membuat laporan bagian ringkasan
24	23/11/2021	18220084	Dewa Ayu Mutiara K P D	Membuat command roll, command endturn, dan main program,
25	24/11/2021	18220059	Faiza Aqiela Zuma	Update ADT Command bagian skill
26	24/11/2021	18220030	Shafira Putri Azmi Imani	Debug main
27	24/11/2021	18220084	Dewa Ayu Mutiara K P D	Menggabungkan skill dan player, menambahkan beberapa fungsi baru agar bisa dijalankan di main program lalu dipush ke github. Merevisi main program dan membuat command buff dan skill lalu dipush ke github.
28	24/11/2021	18220032	Rifki Kaida	Debug undo
29	24/11/2021	18220032	Rifki Kaida	Menyusun laporan bagian ADT Stack

30	25/11/2021	18220059	Faiza Aqiela Zuma	Debug command bagian skill, update banner interface, update dan kalibrasi user interface & banner
31	25/11/2021	18220084	Dewa Ayu Mutiara K P D	Membuat fungsi untuk sorting peringkat pemain di akhir game lalu dipush ke github. Merevisi dan menyesuaikan ADT array lalu dipush ke github.
32	26/11/2021	18220030	Shafira Putri Azmi Imani	Update laporan bagian ringkasan, membuat laporan bagian ADT Array, Mesin Karakter dan Mesin Kata, List Linier, Map, Player, Teleporter
33	26/11/2021	18220047	Dhiya Risalah Ghaida	Menyusun laporan bagian deskripsi program utama
34	22/11/2021	18220084	Dewa Ayu Mutiara K P D	Membantu penyusunan laporan dan memberikan ide bagian fitur tambahan dan algoritma menarik. Menambahkan asumsi pada program utama.
35	26/11/2021	18220059	Faiza Aqiela Zuma	Update undo (tapi masih error), update + rapihin main program, dan update fitur tambahan program.
36	26/11/2021	18220084	Dewa Ayu Mutiara K P D	Mengecek apakah program jalan di linux, lalu melakukan debug dan di-push ke github. Mengupdate command dan dipush ke github.
37	27/11/2021	18220030	Shafira Putri Azmi Imani	Menambahkan laporan bagian ADT.
38	27/11/2021	1820059	Faiza Aqiela Zuma	Debug main program dan test case semua kasus, mengupdate laporan di bagian ADT dan fitur tambahan. Memberi ide mengenai algoritma menarik di laporan.
39	27/11/2021	18220084	Dewa Ayu Mutiara K P D	Membuat UNDO, merevisi ADT Stack, dan merevisi peringkat pemain lalu di push ke github. Mengupdate command, console, dan stack lalu dipush ke github.
40	27/11/2021	1822032	Rifki Kaida	Menyusun laporan bagian ADT Roll dan Skill
41	27/11/2021	18220047	Dhiya Risalah Ghaida	Menyusun laporan bagian algoritma - algoritma menarik

42	28/11/2021	18220084	Dewa Ayu Mutiara K P D	Merapikan folder, memperbaiki include, menyelesaikan undo, melakukan debug secara keseluruhan, lalu dipush ke github. Membuat driver map.
43	28/11/2021	18220030	Shafira Putri Azmi Imani	Membuat driver player, driver array, driver stack, memberi comment pada ADT player
44	28/11/2021	18220032	Rifki Kaida	Update ADT Stack dan ADT Teleporter
45	28/11/2021	18220047	Dhiya Risalah Ghaida	Membuat driver mesin kata
46	28/11/2021	18220047	Dhiya Risalah Ghaida	Revisi beberapa deskripsi pada laporan, mengecek pemakaian PUEBI, dan merapikan format (finalisasi)
47	28/11/2021	18220059	Faiza Aqiela Zuma	Mengupdate laporan di bagian ringkasan, Merapikan driver dan comment ADT Skill, ADT Roll, ADT List Finalisasi Semua Program, mengetes semua case yang ada.