

LAPORAN TUGAS BESAR
IF2111/Algoritma dan Struktur Data


PERMAINAN MOBITANGGA

Dipersiapkan oleh:

Kelompok 10

Muhammad Rifqi Riansyah M.	18220005
Farhandika Zahrir Mufti G.	18220015
Kafi Irgie Rahmansyah	18220020
Kirana Shely Sefiana	18220036
Muhamad Fariz Ramadhan	18220081

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

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		IF2111-TB-10		16
		Revisi	0	28 November 2021

Daftar Isi

Ringkasan	2
Penjelasan Tambahan Spesifikasi Tugas	4
Spesifikasi Pemain 2-4	4
Spesifikasi Skill Mesin Waktu	4
Spesifikasi Skill Baling-Baling Jambu	4
Struktur Data (ADT)	5
ADT Mesin Karakter	5
ADT Mesin Kata	5
ADT Konfigurasi	5
ADT Player	6
ADT Skill	6
ADT Ronde	7
Program Utama	8
Algoritma-Algoritma Menarik	10
Insertion Sort	10
Undo	10
Pembagian Kerja dalam Kelompok	11
Lampiran	12
Deskripsi Tugas Besar	12
Notulen Rapat	12
Log Activity Anggota Kelompok	15
Penjelasan Tambahan terkait Struktur Program	16

1 Ringkasan

Permainan Mobitangga merupakan sebuah permainan nostalgia masa kecil yaitu permainan *board game* yang berbasis dari permainan ular tangga. Permainan ini dapat dilakukan oleh 2 hingga 4 pemain yang berlomba-lomba untuk mencapai garis *finish* yaitu petak ke-N di dalam papan satu dimensi. Berbeda dengan permainan ular tangga pada umumnya, Permainan Mobitangga ini tidak hanya berusaha untuk mencapai garis *finish* tercepat, tetapi juga di dalam perjalanannya para pemain akan mendapatkan *teleporter* dan *skill* yang dapat digunakan untuk membantu pemain dalam mencapai garis *finish* maupun untuk mencegah pemain lain mencapai garis *finish*. Permainan akan dihentikan jika permainan sudah mendapatkan pemenang, yaitu pemain yang mencapai garis *finish* pertama. Peringkat pemain lain akan dilihat berdasarkan posisi terakhir pemain tersebut saat pemenang mencapai garis *finish*.

Berdasarkan deskripsi permainan tersebut, kami membuat program permainan dengan berbasis CLI (*command-line interface*) yang dibuat dalam bahasa C dengan struktur data yang sudah kami pelajari selama perkuliahan. *Game mechanics* yang kami buat terdiri dari 8 komponen, yaitu main menu, peta, *roll*, *teleporter*, *skill*, *buff*, *command*, dan *game flow*.

Laporan mencakup deskripsi umum persoalan *game*, penjelasan tambahan spesifikasi tugas, penjelasan struktur data yang digunakan, penjelasan program utama, algoritma-algoritma menarik, dan pembagian tugas serta lampiran berisi deskripsi tugas, notulen rapat, *log activity* anggota kelompok, dan lampiran lainnya.

Game ini dimulai dengan menampilkan menu yang memberikan pilihan kepada pemain untuk *start game* atau *exit*. Ketika permainan dimulai, seluruh pemain akan berada pada posisi petak ke-1 pada peta dan selanjutnya masing-masing pemain akan melakukan aksi (*command*) secara bergiliran. Pemain dalam setiap gilirannya dapat melakukan aksi dengan memberikan masukan berupa bilangan 0 sampai 7 dengan rincian menu, *exit game*, *roll dice*, *use skill*, *show buff*, *inspect map*, *show position*, *undo*, dan *end turn*. Permainan akan terus berlangsung bergiliran antara semua pemain hingga salah satu pemain mencapai garis *finish* dan menjadi pemenang. Hasil kelompok kami, program sudah berjalan sesuai dengan spesifikasi. Bonus yang berhasil dibuat oleh kelompok adalah jumlah pemain hingga 4, *skill* mesin waktu, dan *skill* baling-baling jambu.

2 Penjelasan Tambahan Spesifikasi Tugas

Kelompok kami mengerjakan dan membuat beberapa fitur bonus, yaitu permainan mampu dimainkan hingga 4 orang serta pemain dapat memiliki *skill* mesin waktu dan baling-baling jambu. Berikut adalah penjelasan dari spesifikasi bonus tersebut.

2.1 Spesifikasi Pemain 2-4

Kami mengimplementasikan permainan Mobitangga yang mampu dimainkan oleh 2 hingga 4 pemain. Permainan dimulai dengan menerima masukan berapa banyak pemain. Kemudian program akan meminta pengguna memasukkan nama masing-masing pemain. Selanjutnya, tiap pemain akan bergantian untuk melakukan giliran sesuai dengan urutan memasukkan nama pemain pada saat awal permainan.

2.2 Spesifikasi Skill Mesin Waktu

Kami mengimplementasikan *skill* Mesin Waktu dalam permainan Mobitangga ini. Penggunaan skill bonus ini memanfaatkan ADT Skill. Saat menggunakan *skill* Mesin Waktu, pemain dapat memilih *opponent* untuk dimundurkan beberapa langkah. Pertama, akan ditampilkan berapa langkah seorang pemain dapat membuat pemain lawan bergerak mundur. Kemudian, pemain diberikan *list opponent* yang ingin dimundurkan N langkah dan pemain diminta untuk memilih pemain lawan mana yang ingin dimundurkan. Jika pemain lawan dapat bergerak mundur sebanyak N langkah, maka pemain lawan tidak mundur dan tidak mengaktifkan teleport di tempat opponent tersebut diam, serta *skill* selesai digunakan. Namun jika pemain lawan dapat bergerak mundur, maka ia akan mundur dan mengaktifkan *teleporter* jika ada, serta *skill* selesai digunakan. Pada kasus bertemu *teleporter*, pemain lawan dapat menggunakan *buff* Imunitas Teleport jika ada.

Dalam pengimplementasian *skill* ini, kami menggunakan fungsi fungsi seperti *getDiceValue* untuk meng-generate angka mata dadu, *playerOption* untuk menerima input dari pemain, *isPlayerCanMove* untuk menentukan apakah pemain lawan dapat bergerak mundur sebanyak N langkah, dan *playerTeleport* untuk mengaktifkan teleport jika ada.

2.3 Spesifikasi Skill Baling-Baling Jambu

Kami mengimplementasikan *skill* Baling-Baling Jambu dalam permainan Mobitangga ini. Penggunaan skill bonus ini memanfaatkan ADT Skill dan secara umum mirip dengan spesifikasi skill mesin waktu. Hal yang membedakan adalah apabila pemain menggunakan *skill* ini maka pemain dapat menggerakkan pemain lawan maju sebanyak N langkah, bukan bergerak mundur. Bilangan N ini nantinya akan diperoleh dari proses *random* angka dadu yang digunakan.

3 Struktur Data (ADT)

Kami menggunakan ADT dasar yang diwajibkan pada Tugas Besar ini seperti *array*, mesin kata, mesin karakter, *stack*, dan *list* (struktur berkait). Kami juga mengimplementasikan beberapa ADT eksternal berbasis pada ADT dasar tersebut seperti ADT Konfigurasi yang memanfaatkan *array*, mesin kata, dan mesin karakter, ADT Player yang memanfaatkan *array*, ADT Skill yang memanfaatkan *list linier*, dan ADT Ronde yang memanfaatkan *stack*.

3.1 ADT Mesin Karakter

- Sketsa struktur data : ADT yang dapat membaca suatu file atau input per-karakter. Kami menggunakan ADT ini sebagai dasar untuk memudahkan pembacaan input. Fungsi START() merupakan fungsi untuk memulai pembacaan dan ADV() untuk melanjutkan proses pembacaan dari mesin karakter.
- Persoalan yang diselesaikan : ADT mesin kata.
- Alasan pemilihan : Dengan ADT ini kami bisa membuat ADT Mesin Kata dan ADT Konfigurasi.
- Diimplementasikan sebagai ADT mesin karakter dengan nama file “mesin_kar.c” dengan file header “mesin_kar.h”.

3.2 ADT Mesin Kata

- Sketsa struktur data : Struktur data lanjutan dari Mesin Karakter. Dalam struktur data ini terdapat 2 prosedur penting yaitu, STARTKATA() untuk mulai membaca kata dan ADVKATA() untuk melanjutkan proses membaca kata.
- Persoalan yang diselesaikan : ADT Konfigurasi
- Alasan pemilihan : ADT ini kami gunakan untuk membantu proses konfigurasi file yang diimplementasikan pada ADT Konfigurasi.
- Diimplementasikan sebagai ADT mesin kata dengan nama file “mesin_kata.c” dengan file header “mesin_kata.h”.

3.3 ADT Konfigurasi

- Skema struktur data : ADT Konfigurasi merupakan ADT yang memproses file konfigurasi untuk dibaca sebagai Peta. ADT ini memanfaatkan ADT mesin karakter dan mesin kata untuk membaca file konfigurasi dan langsung menginput hasilnya ke dalam array peta. Saat proses memasukkan peta, struktur data ini juga langsung mengeset nilai MaxRoll, panjang peta, banyak teleporter, serta letak teleporternya. ADT konfigurasi juga memproses command ‘Map’ untuk menampilkan peta dan posisi player, serta memproses command ‘Inspect’ untuk mengetahui informasi mengenai suatu petak X.
- Persoalan yang diselesaikan : set: nilai MaxRoll, panjang peta, banyak dan letak teleporter, dan menampilkan peta, posisi player, informasi suatu petak.
- Alasan pemilihan : Untuk memudahkan konfigurasi antara player, map, dan teleporter.
- Diimplementasikan sebagai ADT Konfigurasi dengan nama file “Configuration.c” dengan file header “Configuration.h”

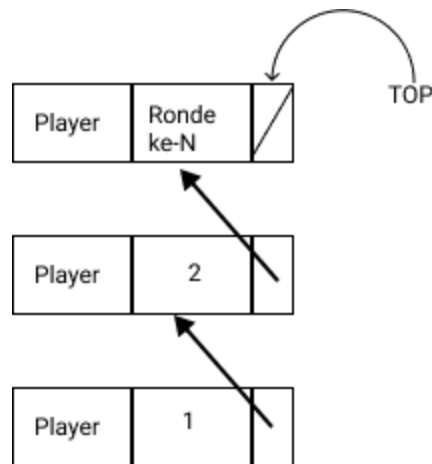
3.4 ADT Player

- Sketsa struktur data : ADT Player merupakan ADT yang menyimpan informasi terkait pemain atau *player*. ADT Player memiliki struktur bernama *player* yang terdiri atas *position* (integer), *name* (array of char), *skills* (skill), dan *buff* (array of boolean). Struktur ini digunakan untuk mengakses semua info pemain secara berurutan, yaitu letak posisi pemain dalam peta, nama pemain, skill-skill yang dimiliki player, dan kondisi buff pemain. *Prototype* terdiri atas beberapa fungsi seperti kreator untuk membentuk player, mengatur dadu player, fungsi-fungsi untuk teleport, serta info buff dari player.
- Persoalan yang diselesaikan : informasi posisi pemain, informasi queue skill pemain, informasi buff pemain.
- Alasan pemilihan : ADT Player digunakan untuk menyimpan segala komponen informasi yang berkaitan dengan pemain tersebut.
- Diimplementasikan sebagai ADT player dengan nama file “Player.c” dengan file header “Player.h”

3.5 ADT Skill

- Skema struktur data : ADT yang dapat meng-generate sebuah skill secara random, kami menggunakan integer untuk meng-*assign* setiap skill (e.g. Skill 1 = Pintu gak kemana mana dst.). Skill tersebut kemudian disimpan dalam sebuah List Skill dengan maksimal 10 skill. Prototype yang digunakan mirip dengan list linier yaitu, konstruktor skill, count skill, getRandomSkill, insert skill, delete skill, print skill, dan get id skill.
- Persoalan yang diselesaikan: Kami menggunakan ADT ini untuk menyelesaikan spesifikasi Skill yang terdapat dalam tugas.
- Alasan pemilihan: Kami memilih ADT ini karena stack memudahkan proses penggunaan skill yaitu menggunakan prinsip list linear untuk menyimpan skill ke dalam list skill yang dimiliki masing-masing player.
- Diimplementasikan sebagai ADT Ronde dengan nama file “Skill.c” dengan file header “Skill.h”. Untuk kegunaan dari masing-masing skill diimplementasikan pada fungsi “useSkill1” sampai “useSkill8” pada file “game.c”

3.6 ADT Ronde



Gambar sketsa ADT Ronde

- Skema struktur data : ADT yang menyimpan state atau posisi player dalam setiap ronde. Setiap ronde kemudian disimpan dengan mengikuti prinsip stack dengan menggunakan pointer. Prototype yang digunakan sangat mirip dengan ADT Stack yaitu mempunyai konstruktor, alokasi, dealokasi, push, dan pop.
- Persoalan yang diselesaikan: Kami menggunakan ADT ini untuk menyelesaikan spesifikasi UNDO yang terdapat dalam tugas.
- Alasan pemilihan: Kami memilih ADT ini karena stack memudahkan proses UNDO dengan proses yang simpel yaitu melakukan pop untuk UNDO.
- Diimplementasikan sebagai ADT Ronde dengan nama file “Round.c” dengan file header “Round.h”

4 Program Utama

Program utama dibagi menjadi beberapa bagian, yang mana proses interaksi antara program dan pengguna terjadi pada *layer view*. Sebelum masuk ke dalam *view*, pengguna masuk ke *main program*. Di *main program* ini pengguna akan bertemu dengan *Menu View* yang akan meminta user untuk memilih apakah ingin memulai permainan atau keluar dari program. Jika user memilih untuk keluar, program akan berpindah ke *Exit View* yang mengatasi proses keluarnya pengguna dari program ini. Jika pengguna tidak memilih untuk keluar, pengguna akan berpindah ke *Game View*. Jika pengguna memilih untuk memulai permainan, pengguna akan diminta untuk memasukan jumlah pemain yang akan bermain, memasukan nama file konfigurasi, lalu mulai bermain. *State* permainan awal ini akan disimpan terlebih dahulu ke dalam ADT *Ronde* yang menggunakan struktur data *stack* dengan prosedur *push*.

Setelah semua proses persiapan selesai, semua pemain akan berada di petak 1 pada *map* dan permainan akan dimulai dengan ronde 1 oleh pemain 1. Selama belum ada pemain yang memenangkan permainan ini atau tidak terjadi kondisi lainnya yang menyebabkan permainan berakhir, proses looping akan terus terjadi. Di dalam looping utama ini terdapat looping kembali yang baru akan berhenti setelah satu ronde telah terselesaikan atau semua pemain telah melakukan gilirannya. Di awal looping ronde ini, setiap pemain memulai gilirannya, *buff* pemain akan direset dan pemain akan mendapatkan 1 *skill* secara acak dari beberapa *skill* yang tersedia. Permainan ini menyediakan 8 *skill* yang dapat dimiliki oleh tiap pemain, yaitu Pintu Ga Ke Mana Mana, Mesin Waktu, Baling Baling Jambu, Cermin Pengganda, Senter Pembesar Hoki, Senter Pengecil Hoki, Mesin Penukar Posisi, dan Teknologi Gagal. Kemudian, program juga akan menampilkan posisi para pemain saat ini dalam *map* yang digunakan.

Selanjutnya, di dalam looping ronde ini terdapat juga looping kembali yang baru akan berhenti setelah satu pemain menyelesaikan gilirannya. Di dalam looping giliran ini, program akan menampilkan aksi apa saja yang dapat dilakukan oleh pemain dan pemain diminta untuk memilih aksi-aksi tersebut. Pemain dapat melakukan aksi dengan memberikan masukan berupa bilangan 0 sampai 7 dengan aksi yang dilakukan berturut-turut sebagai berikut, *exit game*, *roll dice*, *use skill*, *show buff*, *inspect map*, *show position*, *undo*, dan *end turn*.

Jika pemain memilih melakukan *exit game* maka permainan akan diberhentikan secara langsung dengan keluar dari looping utama dan pengguna akan dibawa ke menu utama.

Jika pemain memilih melakukan *roll dice* maka akan dilakukan pemeriksaan terlebih dahulu apakah pemain sudah bergerak atau belum. Aksi ini hanya akan bisa berjalan apabila pemain belum bergerak. Jika pemain belum bergerak, proses akan dilanjutkan dengan memanggil prosedur *playerRollDice* untuk menjalankan aksi *roll dice* tersebut.

Jika pemain memilih melakukan *use skill* maka akan dilakukan pemeriksaan terlebih dahulu apakah pemain sudah bergerak atau belum. Aksi ini hanya akan bisa berjalan apabila pemain belum bergerak. Jika pemain belum bergerak, proses akan dilanjutkan dengan memanggil prosedur *playerUseSkill* untuk menjalankan aksi *use skill* tersebut.

Jika pemain memilih melakukan *show buff* maka program akan memanggil prosedur *showPlayerBuff* yang akan menampilkan status *buff* apa saja yang sedang dimiliki pemain tersebut saat itu.

Jika pemain memilih melakukan *inspect map* maka program akan meminta pemain memasukkan petak mana yang ingin dilakukan *inspect*. Selanjutnya, angka petak yang dimasukkan akan diperiksa menggunakan prosedur *inspectMap* apakah petak kosong, petak terlarang, atau petak dengan teleporter.

Jika pemain memilih melakukan *show position* maka akan ditampilkan di mana posisi semua pemain saat itu dalam *map* yang digunakan dengan memanggil prosedur *showPlayerPosition*.

Jika pemain memilih melakukan *undo* maka akan diperiksa terlebih dahulu saat ini pemain berada dalam ronde ke berapa. Apabila saat ini pemain berada dalam ronde 1, maka program akan keluar dari looping giliran dan looping ronde, kemudian masuk ke bagian awal looping utama dan mengakses info *top* dari stack ronde, yaitu *state game* dari ronde sebelumnya, yang pada kasus ini adalah *state game* awal (sesaat setelah membaca file konfigurasi serta input jumlah dan nama pemain). Namun, apabila saat ini pemain berada dalam ronde lebih dari 1, pemain akan ditanyakan kembali apakah ingin melakukan *undo* kembali atau tidak hingga mencapai batas melakukan *undo* (saat ronde yang dituju mencapai ronde 1). Jika pemain memilih untuk melakukan *undo* kembali, maka akan dilakukan proses *pop* terhadap *stack* ronde yang ada. Namun jika tidak, proses akan dilanjutkan dengan keluar dari looping giliran dan looping ronde, kemudian masuk ke bagian awal looping utama dan mengakses info *top* dari stack ronde, yaitu *state game* dari ronde sebelumnya.

Jika pemain memilih melakukan *end turn* maka akan dilakukan pemeriksaan terlebih dahulu apakah pemain sudah bergerak atau belum. Aksi ini hanya akan bisa berjalan apabila pemain sudah bergerak. Jika pemain sudah bergerak, program akan keluar dari looping giliran dan permainan dilanjutkan dengan pemain berikutnya.

Selanjutnya, setelah pemain menyelesaikan gilirannya dan keluar dari looping giliran tersebut, akan diperiksa apakah posisi pemain sudah mencapai posisi *finish* atau belum. Jika sudah maka permainan akan berakhir, program akan keluar dari looping utama, dan program akan menampilkan *ranking* para pemain. Namun, jika pemain belum mencapai posisi *finish* maka permainan akan dilanjutkan dengan giliran pemain berikutnya (program masuk ke dalam looping giliran kembali). Setelah semua pemain telah selesai melakukan gilirannya, yang artinya satu ronde telah diselesaikan (program keluar dari looping ronde), data *state game* saat ini akan disimpan dan dilakukan *push* ke *stack* ronde. Akan tetapi, proses *push* ini hanya akan dilakukan apabila sebelumnya pemain tidak memilih aksi *undo*.

Kemudian, setelah *push* data dalam satu ronde telah dilakukan, program akan menuju awal looping utama dan mengakses info *top* dari *stack* ronde untuk digunakan sebagai *state game* awal dari ronde berikutnya. Permainan dimulai kembali dengan masuk ke ronde baru dan dimulai dari pemain pertama. Program masuk ke dalam looping ronde kembali, melakukan reset *buff* pemain, menambahkan 1 *skill* acak kepada pemain, dan melakukan looping giliran kembali untuk menjalankan aksi-aksi pemain. Proses di atas akan terus berlangsung hingga salah satu pemain telah mencapai posisi *finish*.

5 Algoritma-Algoritma Menarik

Pada Tugas Besar ini, kami menggunakan algoritma sorting bernama *insertion sort* untuk melakukan proses *ranking* para pemain setelah permainan selesai. Selain itu, terdapat fitur *undo* yang menurut kami memiliki algoritma yang menarik juga. Berikut adalah penjelasan terkait algoritma-algoritma tersebut.

5.1 Insertion Sort

Algoritma *insertion sort* kami gunakan untuk menyortir ranking para players ketika game selesai, ketika *endGame* bernilai *true*. Algoritma ini kami gunakan dibandingkan dengan algoritma sorting lainnya dikarenakan cukup mudah untuk mengimplementasikannya, berbanding terbalik dengan *quick sort* maupun *bubble sort*. Kami rasa, *insertion sort* sudah cukup untuk menangani kasus ini yang dimana panjang array maksimum adalah 4. Dengan performa yang dimiliki algoritma ini, menyortir hal tersebut dalam waktu cepat bukanlah sebuah masalah karena akan terselesaikan pada kecepatan $O(n)$ dengan maksimum proses akan memakan sebanyak 4 detik.

5.2 Undo

Algoritma undo akan berjalan apabila pemain memilih opsi *undo* pada gilirannya. Alasan algoritma ini dinilai menarik karena algoritma ini memanfaatkan ADT *Ronde* dengan struktur data *stack* yang mana *stack* ronde tersebut tidak akan kosong selama permainan berlangsung. *Stack* ronde akan selalu menyimpan *state game* awal sebelum ronde 1 dimulai.

6 Pembagian Kerja dalam Kelompok

No.	Fitur/ADT	NIM Coder	NIM Tester
1.	Architecture pattern	18220015	18220020
2.	ADT Konfigurasi	18220015	18220036
3.	ADT Mesin Karakter dan Kata	18220015	18220036
4.	ADT Ronde	18220015, 18220005	18220036
5.	ADT Player	18220015, 18220081	18220005
6.	ADT Skill	18220020	18220005
7.	<i>Basic Game Mechanics</i>	18220015	18220081
8.	Fitur <i>roll dice + teleport</i>	18220015	18220020
9.	Fitur <i>use skill</i>	18220020	18220036
10.	Fitur <i>show buff</i>	18220020	18220036
11.	Fitur <i>inspect map</i>	18220036	18220020
12.	Fitur <i>show position</i>	18220036	18220015
13.	Fitur <i>undo</i>	18220015	18220020
14.	Game View	18220015	18220005
15.	Menu View	18220015	18220081
16.	Exit View	18220015	18220081

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.

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.


7.2 Notulen Rapat

Asistensi I

Tanggal : 8 November 2021	Catatan Asistensi:
Tempat : Gmeet	
Kehadiran Anggota Kelompok:	
No NIM Tanda tangan	
1 18220005 	Buff itu kayak gimana kak? - Buff itu gausah jadi concent -> itu kalau misal kita punya buff hoki misal kmu dapat skill pintu ga kemana2 , brrti sekarang kamu dibuff kan jadi kalau kamu dpt teleport kamu bisa pake skill itu atau ngga- dan selama kamu belum make skill itu kamu mempunyai buff. Jadi buff itu sebagai status kamu punya skill Gausah pake ADT, cukup pake boolean aja(?) misal boolean klu punya skill = True, kalau udah kepake jadi = False-> jadi kamu udah gapunya buff lagi
2 18220015 	Untuk command Inspect, apakah akan dideteksi suatu petak menjadi tempat keluar dari petak lain? Tidak perlu, hanya akan mendeteksi petak yang menjadi tempat masuk teleporter saja.
3 18220020 	

<p>4 18220036</p>  <p>5 18220081</p> 	
	<p>Tanda Tangan Asisten:</p> 

Asistensi II

Tanggal : 18 November 2021	<p>Catatan Asistensi:</p> <p>Progress</p> <ul style="list-style-type: none"> - Udah bikin ADT - Buat mesin kata nya bingung dan kendala di konfigurasinya - Bikin konverter dari string ke tiap tipe data. Dilakuin bersamaan dengan pembacaan file konfigurasi. - Mark nya diganti jadi '\n'. <p>Untuk bentuk driver nya dibebasin ngga?</p> <ul style="list-style-type: none"> - Iya dibebasin <p>Skill bonus itu kan ada maxroll ny, itu berarti dirandom di awal aja?</p> <ul style="list-style-type: none"> - Iya <p>Untuk skill senter pembesar/pengecil hoki itu cuma bisa dipake sekali trs dibuang?</p> <ul style="list-style-type: none"> - Sekali setiap ronde. Setiap skill pasti dibuang setelah digunakan, kaya kartu gitu (pake skill → buff aktif → skill aktif → kartu skill hilang)
Tempat : Gmeet	
Kehadiran Anggota Kelompok:	
<p>No</p> <p>NIM</p> <p>Tanda tangan</p> <p>1 18220005</p> <p>Tidak Hadir</p> <p>2 18220015</p> 	

<p>3 18220020</p>  <p>4 18220036</p>  <p>5 18220081</p> 	<p>Nampilin skill kalau berdasar id nya gapapa? - Gabole, usahain urut berdasarkan urutan dapetnya</p> <p>Buff apakah perlu dibuat ADT nya tersendiri? - Ga harus, bisa langsung dimasukin ke ADT pemain</p> <p>Buff itu kan status tiap pemain, berarti dlm 1 ronde pemain bisa punya lebih dari 1 buff (mengaktifkan lebih dari 1 skill)? - Boleh lebih dari 1</p> <p>Implementasi bisa atau tidaknya seorang pemain menggunakan skill yang menghasilkan buff A saat pemain memiliki buff A dibebaskan (boleh bisa make skill boleh engga) → ini maksudnya gimana? - Ini bebasin aja, ga ngaruh juga, sesuaiin sama aturan masing2</p> <p>Command Buff itu untuk menampilkan Buff yg dimiliki 1 pemain atau semua pemain? - Pemain yang lagi jalan aja</p>
	<p>Tanda Tangan Asisten:</p> 

7.3 Log Activity Anggota Kelompok

NIM	Nama
18220005	Muhammad Rifqi Riansyah M.
<ul style="list-style-type: none"> 18/11/2021 : Mengimplementasikan ADT Stack 27/11/2021 : Melakukan testing pada beberapa bagian program, Membantu mengerjakan dokumen ADT Ronde, Melakukan debugging minor. 28/11/2021 : Mengerjakan dokumen ADT Skill, penjelasan spesifikasi bonus skill mesin waktu 	
18220015	Farhandika Zahrir Mufti Guenia
<ul style="list-style-type: none"> 08/11/2021 : Menentukan Architecture Pattern 08/11/2021 : Membuat Views (Tampilan) 11/11/2021 : Membuat ADT Konfigurasi 13/11/2021 : Membuat ADT Ronde 13/11/2021 : Membuat ADT Mesin Kata 15/11/2021 : Membantu Pembuatan ADT Player 17/11/2021 : Membuat mekanik dasar game seperti teleportasi dan dadu 20/11/2021 : Memperbaiki dan Membuat sistem redo ronde 23/11/2021 : Membuat driver konfigurasi 24/11/2021 : Membuat driver mesin karakter dan mesin kata 25/11/2021 : Membantu membuat laporan bagian ADT, Lain-Lain, Algoritma Unik, dan Program Utama 27/11/2021 : Membuat driver configuration, round 	
18220020	Kafi Irgie Rahmansyah
<ul style="list-style-type: none"> 23/11/2021 : Membuat ADT Skill beserta list linier 25/11/2021 : Melakukan debugging program 26/11/2021 : Membuat implementasi skill mesin waktu dan baling-baling jambu 27/11/2021 : Membuat driver list linier dan skill, melakukan debugging program 28/11/2021 : Melengkapi laporan bagian penjelasan spesifikasi bonus, program utama, algoritma menarik (undo), melakukan debugging program 	
18220036	Kirana Shely Sefiana
<ul style="list-style-type: none"> 07/11/2021 : Menghubungi asisten dan menjadwalkan asistensi 17/11/2021 : Membuat ADT Teleport 17/11/2021 : Mengerjakan command Map 	

<ul style="list-style-type: none"> • 17/11/2021 : Mengerjakan command Inspect • 27/11/2021 : Mengerjakan laporan bagian penjelasan struktur ADT • 27/11/2021 : Membuat driver mesin karakter, mesin kata, player • 28/11/2021 : Melakukan test code pada fitur-fitur program 	
18220081	Muhamad Fariz Ramadhan
<ul style="list-style-type: none"> • 08/11/2021 : Melakukan notulensi asistensi I • 15/11/2021 : Membuat ADT Player • 18/11/2021 : Mengerjakan command Roll • 25/11/2021 : Mengerjakan laporan bagian ringkasan • 25/11/2021 : Merapikan struktur laporan bagian penjelasan tambahan spesifikasi tugas, struktur data ADT. • 27/11/2021 : Merevisi README.md • 28/11/2021 : Merapikan <i>Log Activity</i> 	

7.4 Penjelasan Tambahan terkait Struktur Program

Pada tugas kami, kami menggunakan library string.h untuk menangani banyak hal dikarenakan mesin kata yang kami miliki memiliki problem yang dimana jika digunakan dua kali atau berada di dalam mesin kata kembali, loop pada mesin kata tidak akan berhenti. Penggunaan library string merupakan teknik yang baik, cukup banyak problem yang bisa kami tangani ketika menggunakan library ini.

Kami juga tidak menggunakan ADT Map seperti banyak kelompok lainnya gunakan, menurut kami ADT konfigurasi sudah cukup modular dan logika yang memproses map banyak kami taruh pada presenter layer.

Kami juga sebisa mungkin untuk mengimplementasikan isi dari buku *Clean Architecture: A Craftsman's Guide to Software Structure and Design* yang terkenal di kalangan programmer. Arsitektur yang kami gunakan berbasis pada MVP (Model, View, Presenter) yang dimana model layer direpresentasikan oleh data yang kami simpan pada file ADT, lalu View sebagai tempat dimana interaksi antara user dan program. Sayangnya, kami tidak bisa mengimplementasikan Presenter Layer dengan benar, banyak logic yang kami taruh pada file ADT dan juga View.

Arsitektur tersebut mengutamakan pembagian tugas yang sangat jelas yang dimana hasil dari penerapan tersebut adalah kemudahan dalam melakukan testing karena satu dan lainnya tidak terlalu terkait satu sama lainnya.