LAPORAN TUGAS BESAR
IF1210 Dasar Pemrograman
K10-E

N I M	N A M A
16523109	William Anthony
19623079	Barru Adi Utomo
16523059	Radhitia Syafi Alfardzan
19623019	Michael Ballard Isaiah Silaen
19623259	M. Riyan Rajab Setiawan

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

2024

HALAMAN PERNYATAAN

Kami Selaku

16523109 - William Anthony
19623079 - Barru Adi Utomo
16523059 - Radhitia Syafi Alfardzan
19623019 - Michael Ballard Isaiah Silaen
19623259 - M. Riyan Rajab Setiawan

"Saya menyatakan bahwa saya mengerjakan tugas besar ini dengan sejujur-jujurnya, tanpa menggunakan cara yang tidak dibenarkan. Apabila di kemudian hari diketahui saya mengerjakan tugas besar ini dengan cara yang tidak jujur, saya bersedia mendapatkan konsekuensinya, yaitu mendapatkan nilai E pada mata kuliah IF1210 Dasar Pemrograman Semester 2 2023/2024."

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE

F00_RNG.py
F01_Register.py
F02_Login.py
F03_Logout.py
F04_Help.py
F05_Monster.py
F06_Potion.py
F07_Inventory.py
F08_Battle.py
F09_Arena.py
F10_Shop_Currency.py
F11_Laboratory.py
F12_ShopManagement.py
F13_MonsterManagement.py
F14_Load.py
F15_Save.py
F16_Exit.py
B03_MonsterBall.py
B04_Minigame.py
B05_Peta_Kota_Danville.py

DESKRIPSI PERSOALAN

Pada Tugas Besar IF1210 Dasar Pemrograman 2024 ini, kita diberi tugas untuk menciptakan sebuah game RPG (Role Playing Game) yang bertindak sebagai simulasi permainan Purry si Platypus. Di tengah tantangan yang dihadapi Purry dalam menghadapi Dr. Asep Spakbor, kita sebagai agen-agen harus merancang sebuah pengalaman game yang menarik dan interaktif untuk membantu Purry melawan ancaman tersebut.

Untuk mempersiapkan simulasi game, kita harus memanfaatkan konsep rekayasa array dengan cermat. Langkah pertama yang krusial adalah mengubah data dari format CSV ke dalam kumpulan array yang dapat direkayasa. Hal ini dicerminkan oleh bigdata yang akan memungkinkan kita untuk menyusun strategi pertempuran dan mengelola karakter serta atribut mereka dengan efisien. Setelah pengolahan data, fokus selanjutnya adalah mengembalikan hasil manipulasi array ke dalam format CSV lagi, sehingga memungkinkan simpanan dan pemrosesan data yang lancar.

Kemampuan untuk mengubah data secara efisien, menerapkan fungsi-fungsi rekayasa array, dan memastikan integritas data dalam format yang sesuai akan menjadi kunci kesuksesan dalam menciptakan game RPG yang membantu Purry dalam misinya melawan Dr. Asep Spakbor.

DESAIN PERINTAH

F00_RNG

Input:
Interval → rentang bilangan acak yang diinginkan

Output:
Bilangan acak dalam rentang interval.

F01_Register

Input:
data → array yang berisi data pengguna
monster_data → array yang berisi data monster

Output:
Memperbarui array data pengguna dengan data pengguna baru jika registrasi berhasil menampilkan pesan keberhasilan registrasi dan detail monster yang dipilih.

F02_Login

Input:
(Tidak ada)

Output:
Jika login berhasil, akan mengembalikan nilai True.

Jika login gagal karena username tidak ditemukan, akan mencetak pesan "Username salah" dan mengembalikan nilai False.

Jika login gagal karena password salah, akan mencetak pesan "Password salah" dan mengembalikan nilai False.

F03_Logout

Input:
(Tidak ada)

Output:
Mengembalikan nilai True jika logout berhasil. Mengembalikan nilai False jika logout gagal.

F04_Help

Input:
(Tidak ada)

Output:
Berdasarkan status role yang ada pada sistem, help akan memberikan pesan berupa perintah apa saja yang dapat dilakukan. Untuk setiap role, terdapat perintah uniknya masing-masing.

F05_Monster

Input:
(Tidak ada)

Output:
Level, Defend, and Attack untuk monster

F06_Potion

Input:
(Tidak ada)

Output:
function StrengthPotion, ResiliencePotion, ResiliencePotion untuk battle

F07_Inventory

Input:

bigdata → dictionary yang berisi data besar termasuk inventaris pengguna
user_id → Integer yang merupakan ID pengguna
monster_data → Dictionary yang berisi data monster
monster_level → Integer yang merupakan level monster
id → Input dari pengguna untuk menampilkan detail item

Output:

Menampilkan inventaris pengguna dengan daftar monster dan item, menampilkan detail monster atau item berdasarkan ID yang diberikan oleh pengguna, memperbarui inventaris pengguna jika ada perubahan dan menampilkan pesan keberhasilan atau detail monster yang dipilih.

F08_Battle

Input :

Monster Data → menggunakan bigdata

serang → pengguna diminta untuk melakukan tindakan serang

potion → pengguna diminta untuk menggunakan potion

bola monster → pengguna diminta untuk menggunakan bola monster

keluar → pengguna diminta untuk keluar dari sistem Battle

Output :

Akan ditampilkan pertarungan, muncul gambar ASCII monster lawan beserta statistiknya, disertai pesan kemunculan. Monster pemain juga tampil dengan gambar dan statistik serta pesan "GO, [type]!!". Pemain memilih aksi seperti menyerang, menggunakan potion, atau keluar dari pertarungan. Setiap giliran, tampilan menunjukkan status giliran, hasil serangan, dan HP yang tersisa. Jika menang, muncul pesan kemenangan dan jumlah OC yang diperoleh. Jika kalah atau

keluar, muncul pesan kekalahan atau bahwa pemain meninggalkan pertarungan.

F09_Arena

Input :

monster data → menggunakan data monster yang diambil Menampilkan gambar ASCII dan statistik monster yang dipilih pemain serta monster lawan.

monster choice → pengguna diminta untuk memilih monster yang akan digunakan

keluar → pengguna diminta untuk keluar dari sistem arena

stage → pengguna diminta untuk melanjutkan ke stage berikutnya atau ‘keluar’ dari pertarungan

Output :

Akan ditampilkan gambar ASCII dan statistik monster yang dipilih pemain serta monster lawan. Selama pertarungan, status giliran dan hasil serangan, termasuk damage yang diberikan dan diterima oleh masing-masing monster, ditampilkan. Pada akhir setiap stage, muncul pesan kemenangan atau kekalahan. Setelah semua stage selesai, ditampilkan total hadiah yang diperoleh, jumlah stage yang diselesaikan, serta total damage yang diberikan dan diterima.

F10_Shop_Currency

Input :

lihat → pengguna diminta untuk melihat daftar item yang tersedia

beli → pengguna diminta untuk melihat daftar pembelian item

keluar → pengguna diminta untuk keluar dari sistem Shop_Currency

Output :

Akan ditampilkan tabel rinci saat pengguna melihat item di toko: monster dengan ID, Nama, ATK Power, DEF Power, HP, Stok, dan Harga; potion dengan ID, Tipe, Stok, dan Harga. Saat pembelian, sistem mengkonfirmasi jika berhasil atau memberikan alasan jika gagal. Saat keluar, pengguna menerima pesan perpisahan. Output ini memastikan informasi jelas dan pengalaman yang baik, menggunakan perulangan, list, dan kondisional. Data diperbarui dari file CSV untuk memudahkan belanja dan memberikan umpan balik yang baik.

F11_Laboratory

Input :

big data→ kumpulan seluruh csv
user_id → integer yang mewakili letak data user di user.csv

Output :

Daftar monster yang dimiliki pengguna beserta level mereka. Biaya untuk meningkatkan level monster. Pesan konfirmasi dan hasil dari proses peningkatan level monster.

F12_ShopManagement

Input :

aksi → perintah untuk input berikutnya (lihat/tambah/ubah/hapus/keluar)
tipeshop → tipe barang transaksi yang mau diinteraksi (potion/monster)
targerid → id monster yang mau ditransaksikan
targetstock →jumlah stock yang tersedia
targetprice → harga item yang ditransaksikan
potiontargetid → id potion yang ingin ditransaksikan
ans → verifikasi penghapusan item yang ditransaksikan

Output :

tergantung input aksi :

- lihat : tampilan item-item yang ditransaksikan sesuai input tipeshop (potion/monster)
- tambah : menambah jenis item yang ditransaksikan sesuai input tipeshop (potion/monster.)
- ubah : mengubah jumlah dan/atau harga jenis item yang ditransaksikan sesuai input tipeshop (potion/monster)
- hapus : menghapus jenis item yang ditransaksikan sesuai input tipeshop (potion/monster)

F13_MonsterManagement

Input :

bigdata → Dictionary berisi data besar yang mencakup data monster.

Output :

Menampilkan menu utama dengan opsi untuk menampilkan semua monster, menambah monster baru, atau keluar dan menampilkan semua monster dalam database dan juga menampilkan jika ada terjadinya kesalahan (input tidak valid)

F14_Load

Input :

nama_folder → pengguna diminta untuk mengisi nama folder

Output :

Akan disimpan data csv ke bentuk array dalam folder nama folder yang diinput. Jika folder tidak ditemukan, akan diberikan pesan kesalahan.

F15_Save

Input :

nama_folder → pengguna diminta untuk mengisi nama folder

Output :

Akan disimpan data array ke bentuk csv dalam folder nama folder yang diinput. Jika folder tidak ditemukan, akan diberikan pesan kesalahan.

F16_Exit

Input :
pilihan → pengguna diminta untuk mengisi pilihan (y/n)

Output :
Jika pilihan berupa n, maka tidak akan ada data yang disimpan. Sebaliknya, jika dipilih y maka data akan disimpan dan program akan berakhir.

B03_MonsterBall

input:
user_monster_pool → dictionary yang berisi data monster yang dimiliki pengguna
user_id, monster_id, monster_level → integer yang berupa ID dan level

output:
Menampilkan pesan keberhasilan atau kegagalan dalam menangkap monster dan memperbarui 'user_monster_pool' jiak monster berhasil ditangkap

B04_Minigame

input:
bigdata → dictionary yang berisi data monster, inventory, dan lainnya
user_id → integer dari ID
choice → string yang merupakan pilihan pengguna diminta untuk memilih minigame yang diinginkan ("Y" atau "N")

output:
Menampilkan menu utama dengan daftar item dan menampilkan item yang diperoleh setiap putaran jacckpot dan juga menampilkan jumlah OC yang diperoleh atau pesan jika pengguna mendapatkan tiga item yang sama

B05_Peta_Kota_Danville

input:
name → string yang merupakan nama agen
position → Tuple yang merupakan posisi awal agen (x,y)

map_data → list of list yang berisi data peta
agent_position → Tuple yang merupakan posisi agen (x,y)
direction → string yang merupakan arah pergerakan agen (“W”, “S”, “A”, “D”
action → string yang merupakan tindakan pengguna saat berhenti (“HELP, “LOGOUT”)

output:
Menampilkan peta dengan posisi agen, menampilkan pesan lokasi agen dan apakah agen berada di area khusus, menampilkan pesan hasil pergerakan agen, dan menampilkan pesan tindakan saat terhenti

DESAIN KAMUS DATA

file: user.csv

```
rowuser
     id : integer
     username : string
     password : string
     role : string
     oc : integer

usercsv
     baris : array[1..maxdatarow] of rowuser
     neff : integer[1..maxdatarow]
```

file: monster.csv

```
rowmonster
     id : integer
     type : string
     atk_power : integer
     def_power : integer
     hp : integer

monstercsv
     baris : array[1..maxdatarow] of rowmonster
     neff : integer[1..maxdatarow]
```

file: item_inventory.csv

```
rowitem_inventory
     user_id : integer
     type : string
     quantity : integer

item_inventorycsv
     baris : array[1..maxdatarow] of rowitem_inventory
     neff : integer[1..maxdatarow]
```

file: monster_inventory.csv

```
rowmonster_inventory
     user_id : integer
     monster_id : integer
     level : integer

monster_inventorycsv
     baris : array[1..maxdatarow] of rowmonster_inventory
     neff : integer[1..maxdatarow]
```

file: item_shop.csv

```
rowitem_shop
     type : string
     stock : integer
     price : integer

item_shopcsv
     baris : array[1..maxdatarow] of rowitem_shop
     neff : integer[1..maxdatarow]
```

File: monster_shop.csv

```
rowmonster_shop
     monster_id : integer
     stock : integer
     price : integer

monster_shopcsv
     baris : array[1..maxdatarow] of rowmonster_shop
     neff : integer[1..maxdatarow]
```

F00 - function RNG

```
KAMUS LOKAL
     interval, seed, a, c, m, random_number : integer
```

F01 - function PilihMonster

```
KAMUS LOKAL
    monster_data : list of dict
    pilihan : string
```

```
    b : integer
    monster_id : integer
    monster_type : string
    monster_atk_power : integer
    monster_def_power : integer
    monster_hp : integer
    a : string
```

F01 - function Registrasi

```
KAMUS LOKAL
    data : list of list
    monster_data : list of dict
    usernamePendaftar : string
    passwordPendaftar : string
```

F01 - function PeriksaPassword

```
KAMUS LOKAL
    bigdata : dict
    usernamePendaftar : string
    passwordPendaftar : string
    monster_data : list of dict
    data : list of list
```

F01 - function PeriksaUsername

```
KAMUS LOKAL
    bigdata : dictionary
    usernamePendaftar : string
    passwordPendaftar : string
    monster_data : list of dict
    data : list of list
```

F01 - function PeriksaUsernameUnik

```
KAMUS LOKAL
    data : list of list
    usernamePendaftar : string
```

F01 - procedure TambahAkunBaru

```
KAMUS LOKAL
    bigdata : dictionary
    usernamePendaftar : string
    passwordPendaftar : string
    monster_data : list of dict
    data : list of list
    id_akhir : list
    last_id : integer
    new_id : integer
    monster_id : integer
    monster_type : string
    data_user : dictionary
    data_monster : dictionary
```

F02 - function Login

```
KAMUS LOKAL
    bigdata : dictionary
    user_data : dictionary
    usernamePengguna : string
    passwordPengguna : string
    matriks_user : list of lists
    check_username : boolean
    check_pw : boolean
    user : list
```

F03 - function Logout

```
KAMUS LOKAL
    text_ascii : dictionary
    login_state : boolean
    logout : function
```

F04 - function Help

```
KAMUS LOKAL
    status : dictionary
```

F05 - function MonsterLevel

```
KAMUS LOKAL
     monster_data : dictionary
```

```
     n_times : integer
     attack : integer
     defense : integer
     hp : integer
```

F05 - function Attack

```
KAMUS LOKAL
     monster_atk : integer
     monster_def : integer
     damage : float
```

F05 - function Defend

```
KAMUS LOKAL
     monster_def : integer
     damage_reduction : float
```

F05 - function MonsterDetail

```
KAMUS LOKAL
     bigdata : dictionary
     monster_id : integer
     level : integer
     monster_data : list of dict
     data : dictionary
```

F06 - function PotionMenu

```
KAMUS LOKAL
     user_inventory : list of dict
     monster_data : dictionary
     count : integer
     user_choice : integer
```

F06 - function StrengthPotion

```
KAMUS LOKAL
     monster_data : dictionary
```

F06 - function HealingPotion

```
KAMUS LOKAL
     current_monster_data : dictionary
     monster_data : dictionary
```

F06 - ResiliencePotion

```
KAMUS LOKAL
     monster_data : dictionary
```

F07 - function DisplayMonsterStat

```
KAMUS LOKAL
     monster_data : dictionary
     monster_level : integer
```

F07 - function ShowInventory

```
KAMUS LOKAL
     bigdata : dictionary
     user_id : integer
     inventory : dictionary
     data_monster : list
     data_item : list
     counter : integer
     inventory_2 : list
     banyak_monster : integer
     banyak_item : integer
     i : integer
     id : string
```

F07 - function DetailItem

```
KAMUS LOKAL
     bigdata : dictionary
     item_id : integer
     inventory : list
     banyak_id : integer
     monster_data : dictionary
```

F07 - function InventoryUserDict

```
KAMUS LOKAL
     bigdata : dictionary
```

```
    user_id : integer
    data_monster : list
    data_item : list
    inventory : dictionary
    i : integer
```

F07 - UserInventory

```
KAMUS LOKAL
    user_id : integer
    bigdata : dictionary
    type : string
    data : list
    list_item : list
    i : dictionary
```

F08 - function PrintMonster

```
KAMUS LOKAL
    side : string
    monster_data : dictionary
    monster_level : integer
    status : string
```

F08 - function MonsterAppear

```
KAMUS LOKAL
    monster_pool : dictionary
```

F08 - function Battle

```
KAMUS LOKAL
    bigdata : dictionary
    user_id : integer
    in_arena : boolean
    level_setting : integer
    monster_pool : dictionary
    exit_battle : boolean
    damage_dealt : integer
    damage_received : integer
    monster_appear_id : integer
    monster_appear_level : integer
    monster_appear : dictionary
```

```
    user_monster_pool : list
    user_inventory_pool : list
    id_count : integer
    user_choice : string
    user_monster_level : integer
    user_monster : dictionary
    used_potion : boolean
    turn_count : integer
    win_battle : boolean
    oc_gain : integer
```

F09 - function Arena

```
KAMUS LOKAL
    bigdata : dictionary
    user_id : integer
    total_damage_dealt : integer
    total_damage_received : integer
    stage : integer
    win_battle : boolean
    damage_dealt : integer
    damage_received : integer
```

F10 - function DisplayItems

```
KAMUS LOKAL
    bigdata : dictionary
    item_type : string
    monster_temp : list
    monster_shop : list
    monster : dictionary
    id_temp : integer
    temp_index : integer
    data : dictionary
    item_temp : list
    item_shop : list
    potion : dictionary
```

F10 - function BuyItem

```
KAMUS LOKAL
    bigdata : dictionary
    user_id : integer
```

```
     item_id : integer
     oc : integer
     monster : dictionary
     add_potion : dictionary
     item_index : integer
     quantity : integer
```

F10 - function Shop

```
KAMUS LOKAL
     bigdata : dictionary
     user_id : integer
     oc : integer
     action : string
     item_type : string
```

F11 - function Laboratory

```
KAMUS LOKAL
    bigdata : dictionary
    user_id : integer
    cost : list
    user_oc : integer
    user_monster : list
    monster : dictionary
    choice : integer
    upgrade_cost : iegernt
    confirmation : string
```

F12 - function ShopManagement

```
KAMUS LOKAL
     action : string
     item_type : string
     available_id : list
     monster : dictionary
     monster_id : integer
     monster_stock : integer
     monster_price : integer
     add_monster : dictionary
     confirmation : string
```

```
     item_id : integer
     item_stock : integer
     item_price : integer
     delete_monster : dictionary
```

F13 - function MonsterManagement

```
KAMUS LOKAL
     choice : string
     available_type : list
     monster_type : string
     monster_atk : integer
     monster_def : integer
     monster_hp : integer
     new_monster : dictionary
     confirmation : string
```

F14 - function CariFolder

```
KAMUS LOKAL
     nama_folder : string
     address : string
     valid : boolean
```

F14 - function Load

```
KAMUS LOKAL
     parser : ArgumentParser
     args : Namespace
     address : string
     valid : boolean
     i : integer
     bigdata : dictionary
```

F15 - function tulis_csv

```
KAMUS LOKAL
     data : any
     folder : string
     file : string
     f : file
```

F15 - function save

```
KAMUS LOKAL
     address : string
     i : integer
```

F16 - function exit

```
KAMUS LOKAL
     bigdata : dictionary
     valid : boolean
     simpan : string
```

B03 - function MonsterBall

```
KAMUS LOKAL
     seed : integer
     temp : dictionary
```

B04 - function Minigame

```
KAMUS LOKAL
    choice : integer
```

B04 - function PlayJackpot

```
KAMUS LOKAL
    choice : string
```

B04 - function PlayHangman

```
KAMUS LOKAL
    choice, chosen_word : string
    lives, oc_gain, word_length : integer
    end_of_game : boolean
```

B05 - function PrintMap

```
KAMUS LOKAL
    i, j : integer
```

B05 - function CheckLocation

```
KAMUS LOKAL
    places : dictionary
    adjacent_places : list of str
```

B05 - function MoveAgent

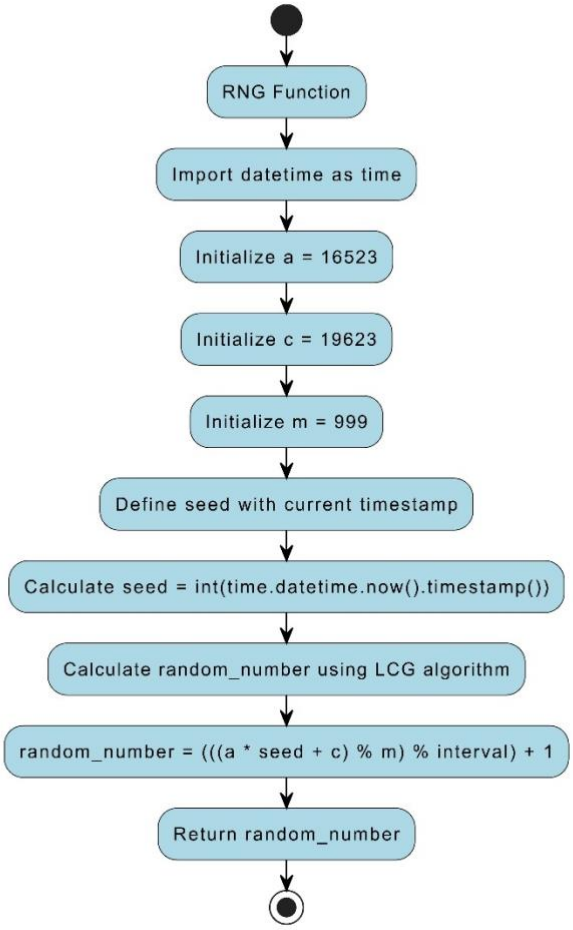
```
KAMUS LOKAL
    x, y : integer
```

B05 - procedure peta_kota_danville

```
KAMUS LOKAL
    map_string : string
    map_data : list of list of str
    location : string
```

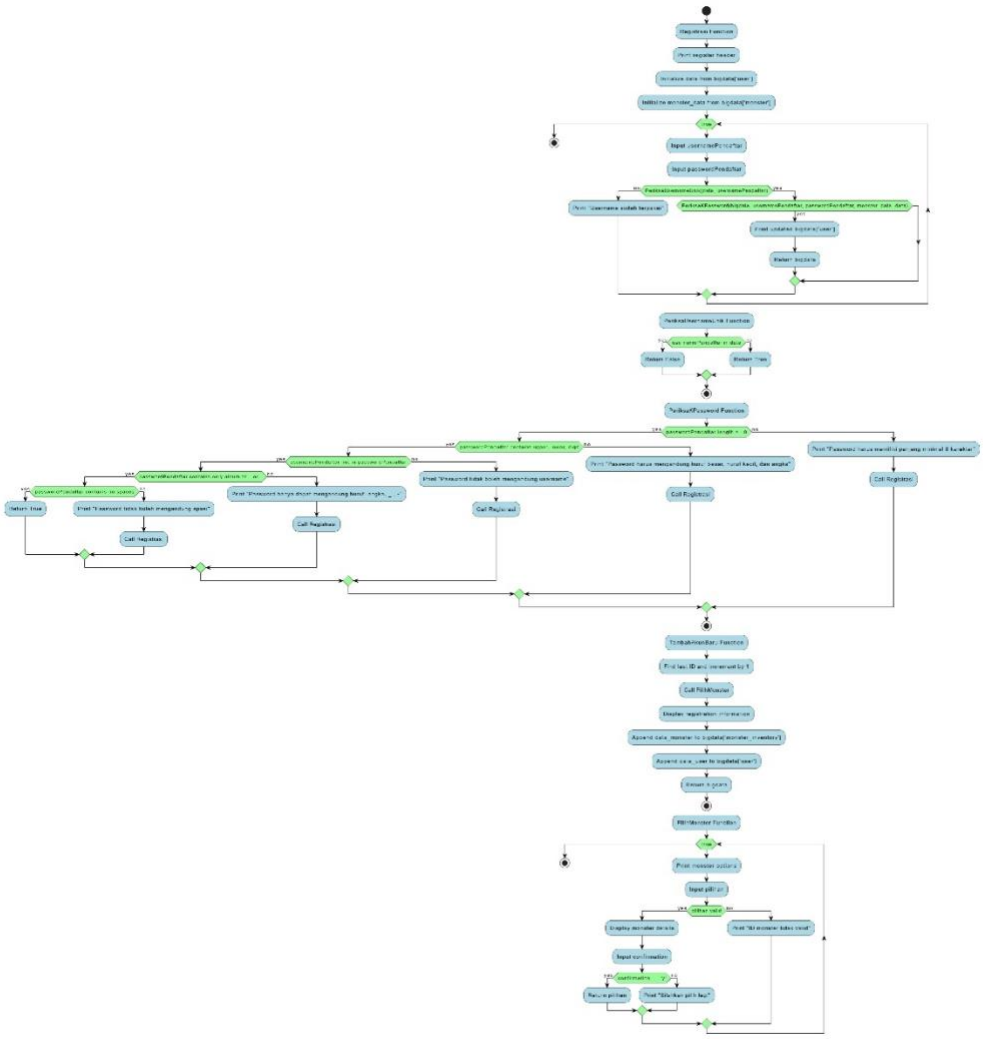
DESAIN DEKOMPOSISI ALGORITMIK DAN FUNGSIONAL PROGRAM

F00_RNG



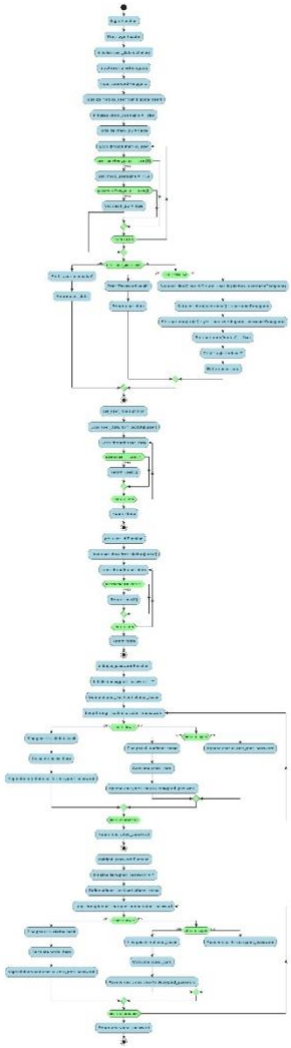
Gambar 1 : F00 – RNG

F01_Registrasi



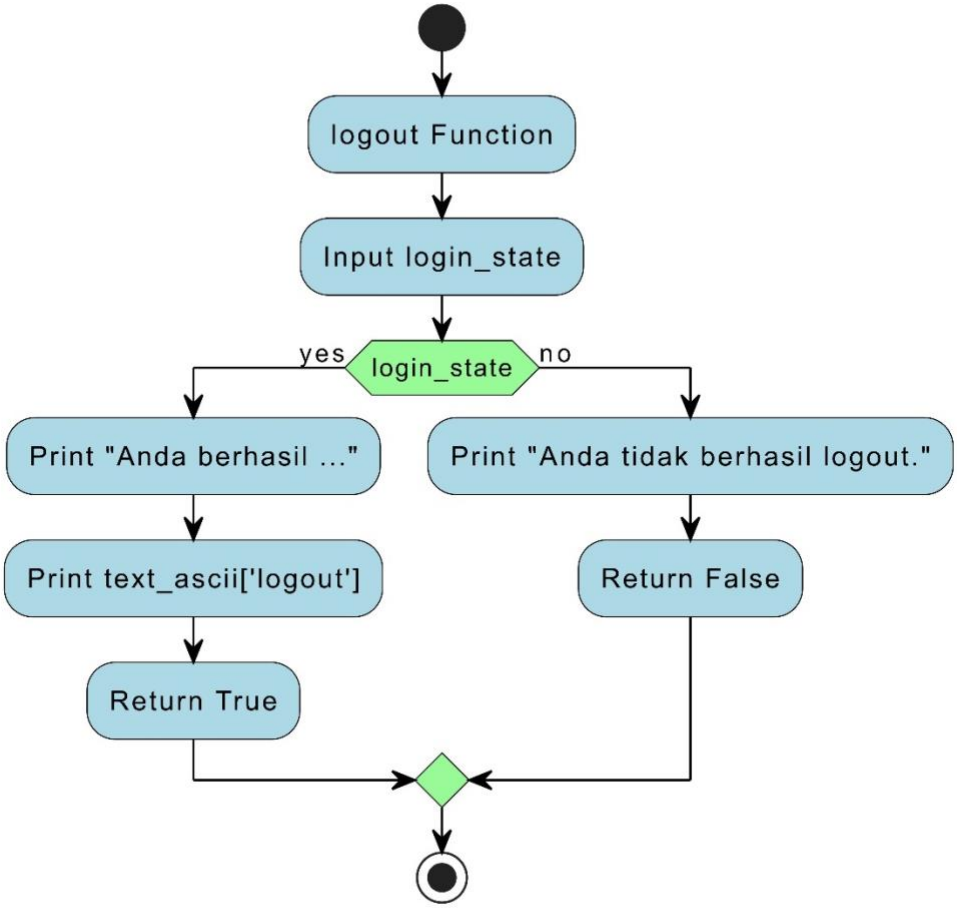
Gambar 2 : F01 - Register

F02_Login



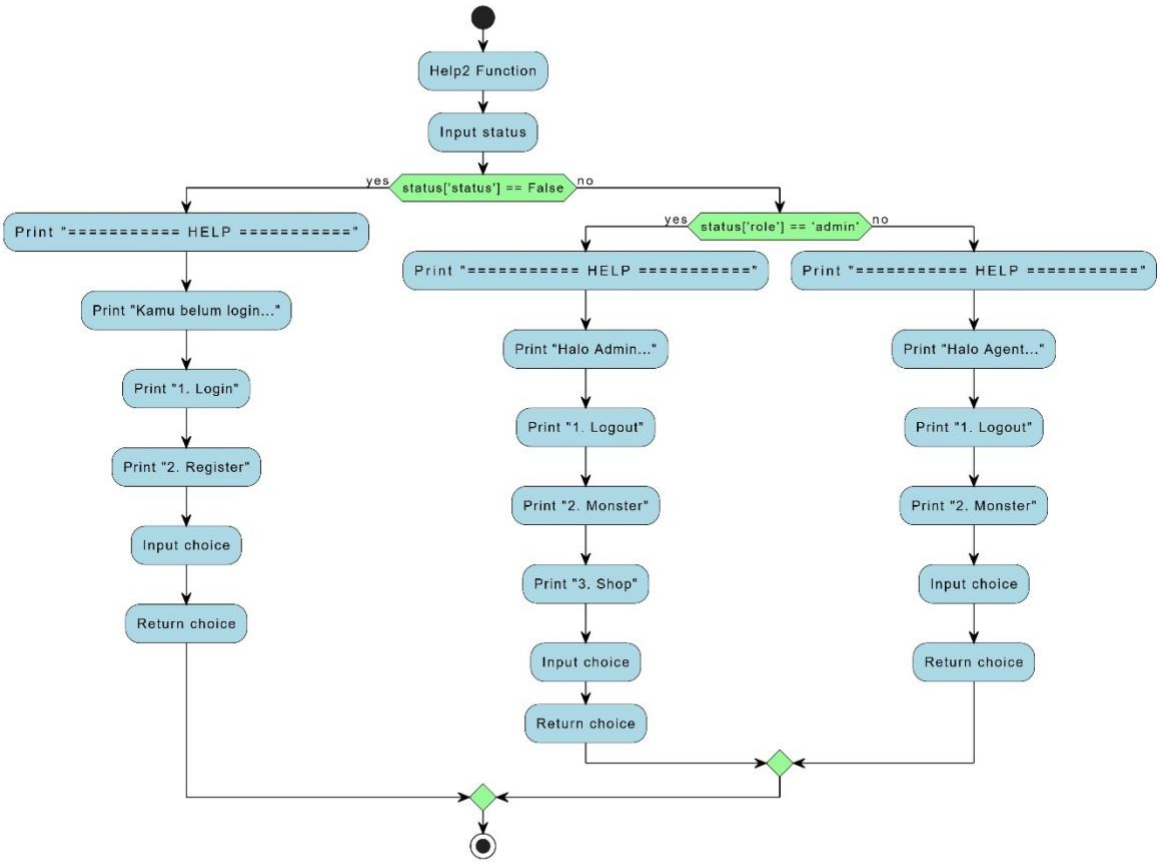
Gambar 3 : F02 - Login

F03_Logout



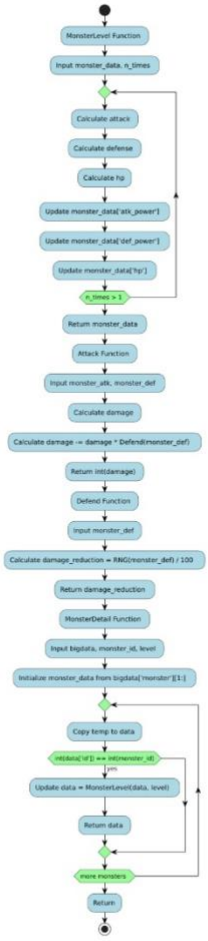
Gambar 4 : F03 - Logout

F04_Help



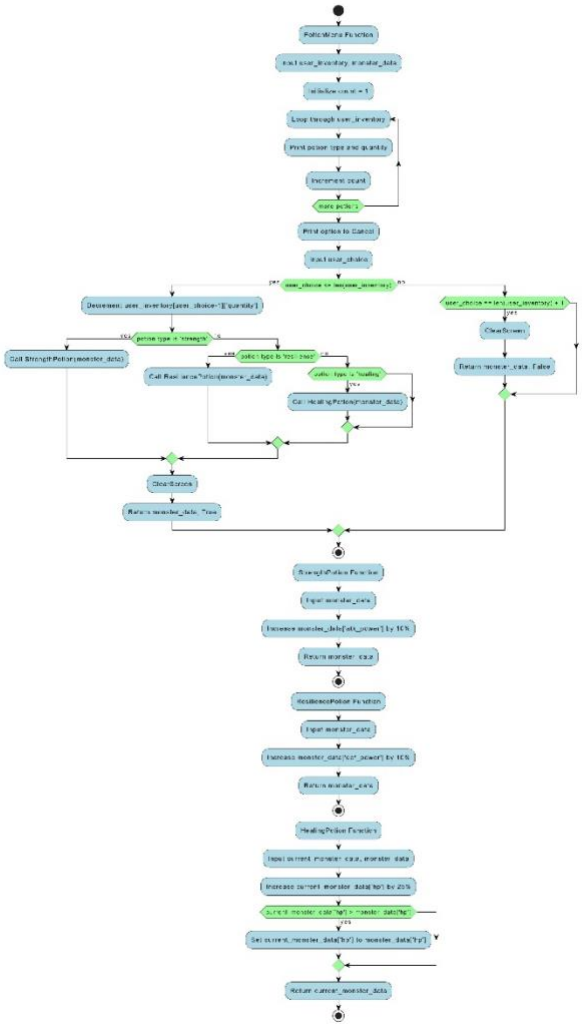
Gambar 5 : F04 – Login

F05_ Monster



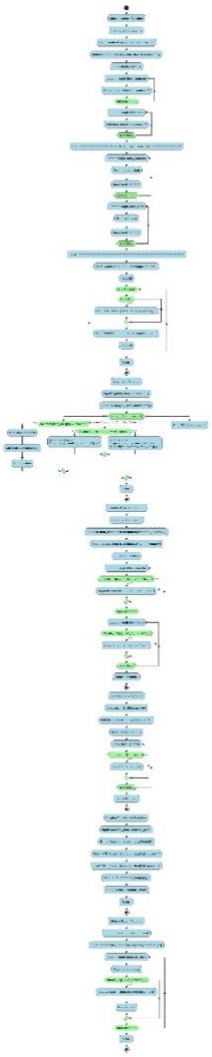
Gambar 6 : F05 - Monster

F06_Potion



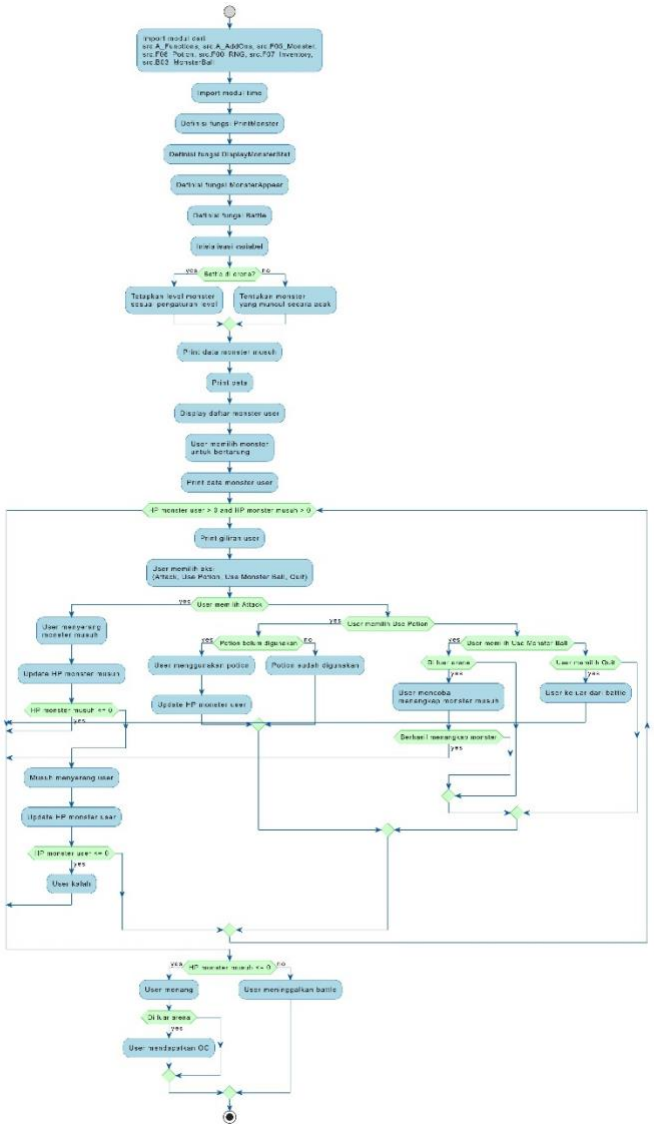
Gambar 7 : F06 - Potion

F07_Inventory



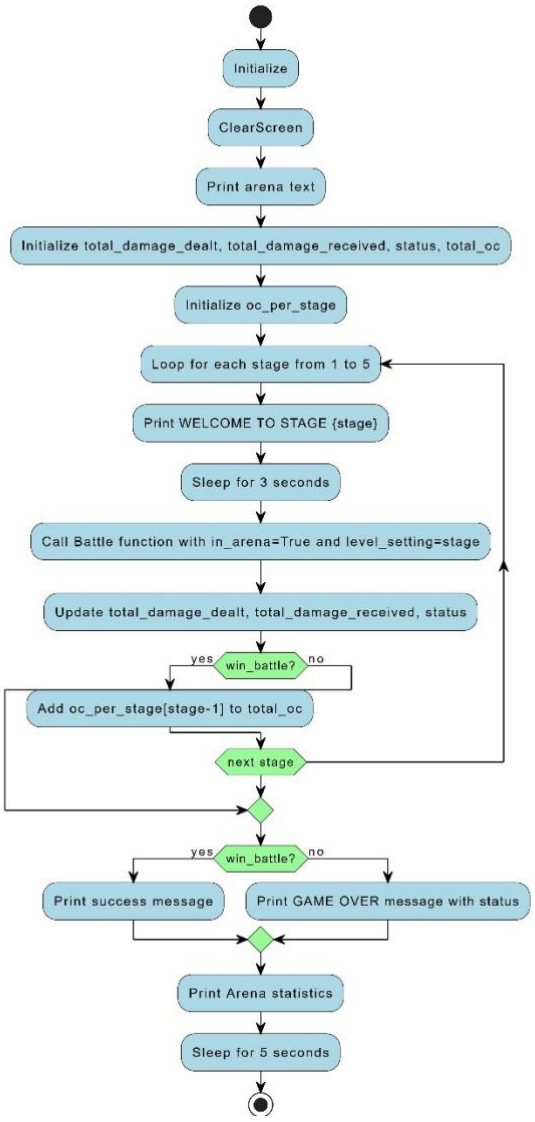
Gambar 8 : F07 - Inventory

F08_Battle



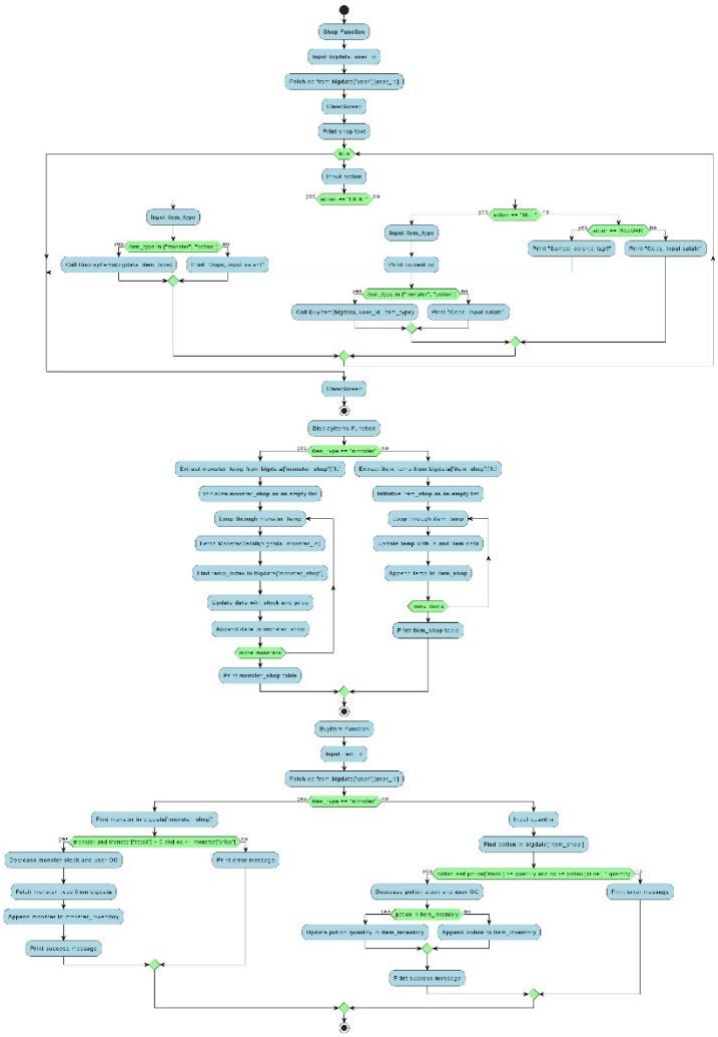
Gambar 9 : F08 - Battle

F09_Arena



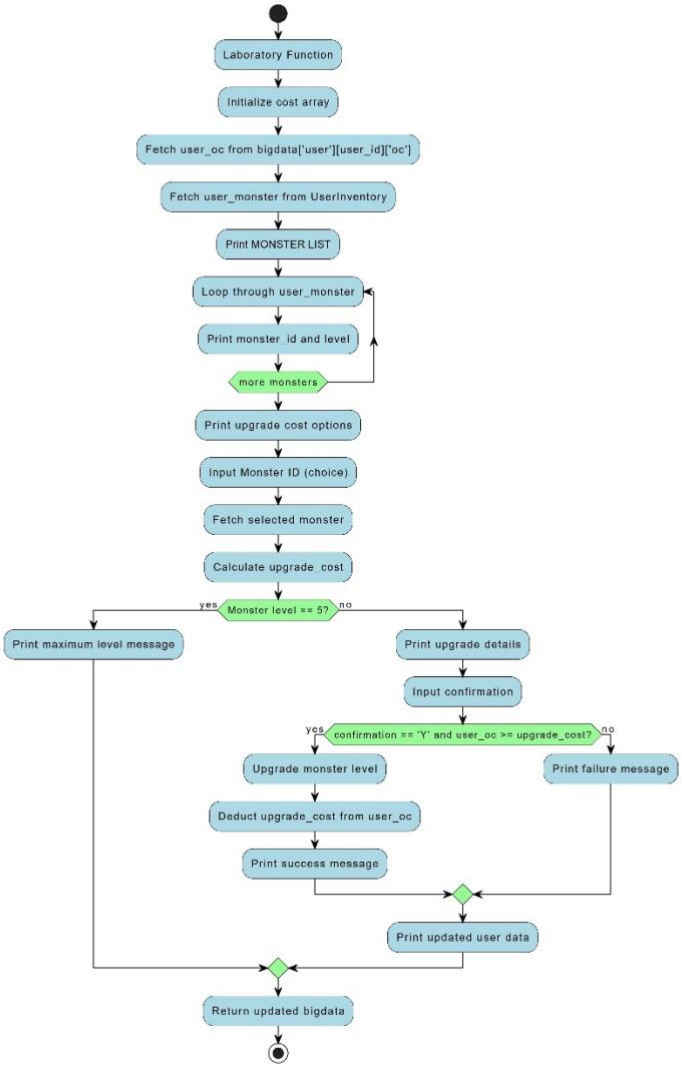
Gambar 10 : F09 - Arena

F10_Shop_Currency



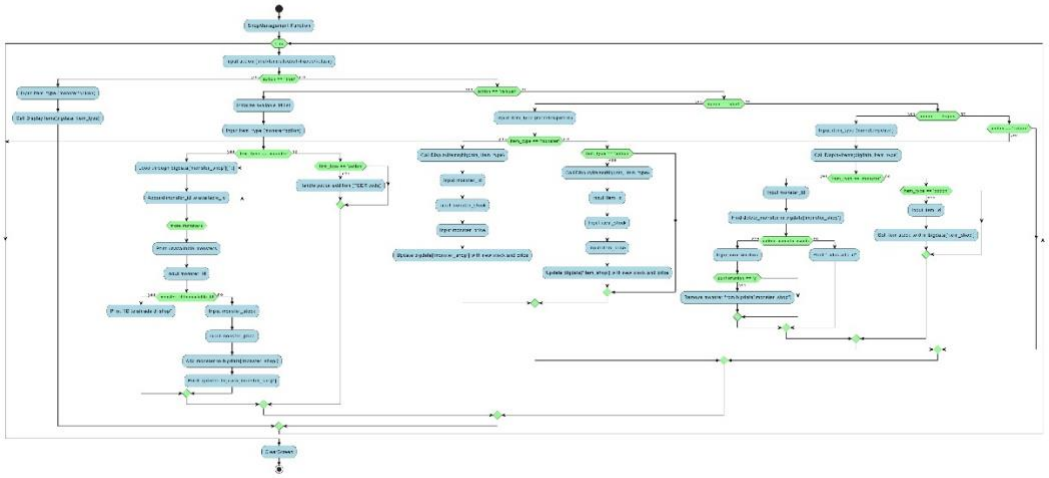
Gambar 11 : F10 - Shop Currency

F11_Laboratory



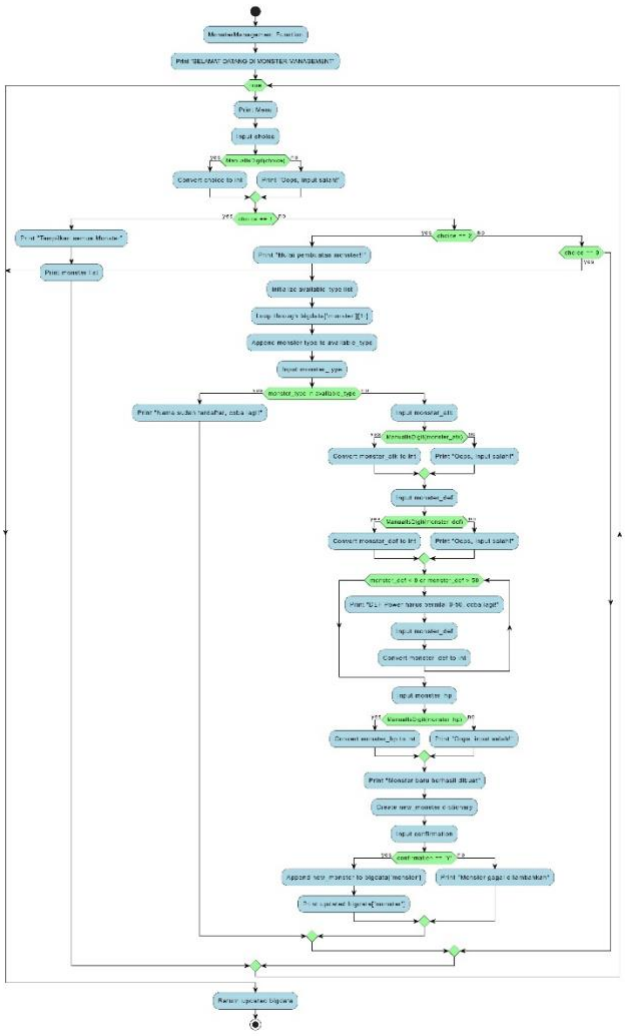
Gambar 12 : F11 - Laboratory

F12_ShopManagement



Gambar 13 : F12 - ShopManagement

F13_MonsterManagement

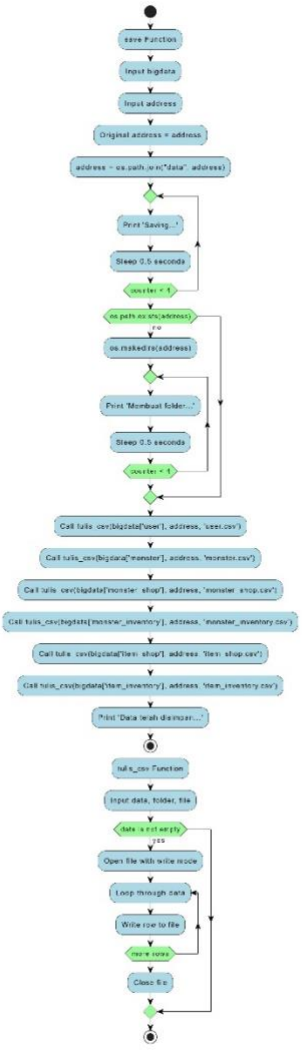


Gambar 14 : F13 - MonsterManagement

F14_Load

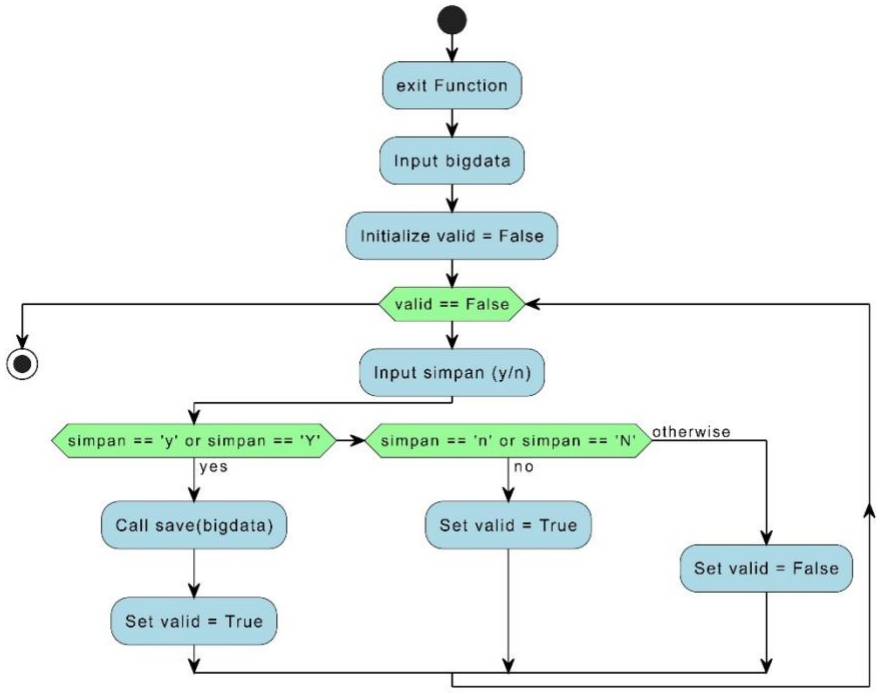
Gambar 15 : F14 - Load

F15_Save



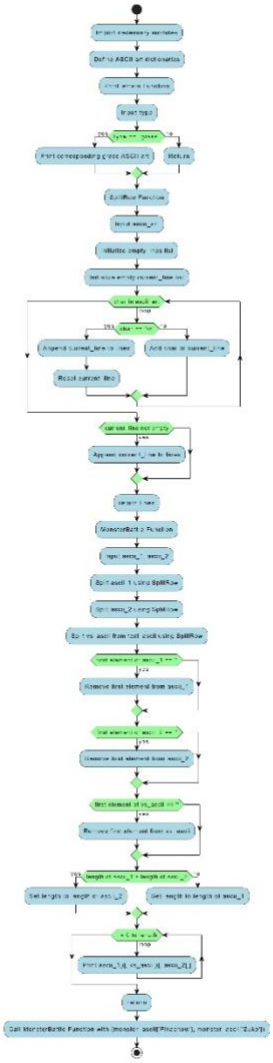
Gambar 16 : F15 - Save

F16_Exit



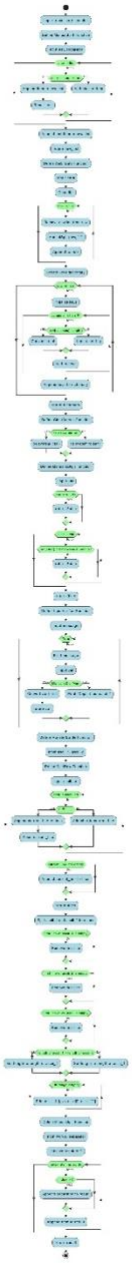
Gambar 17 : F16 - Exit

A_AddOns



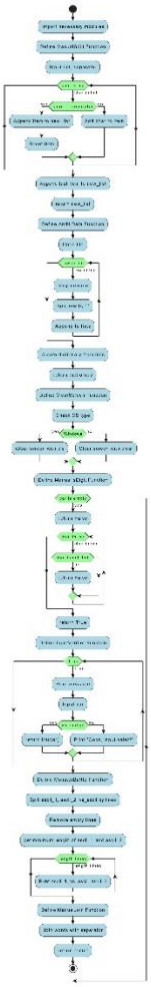
Gambar 18 : A_AddOns

A_Functions



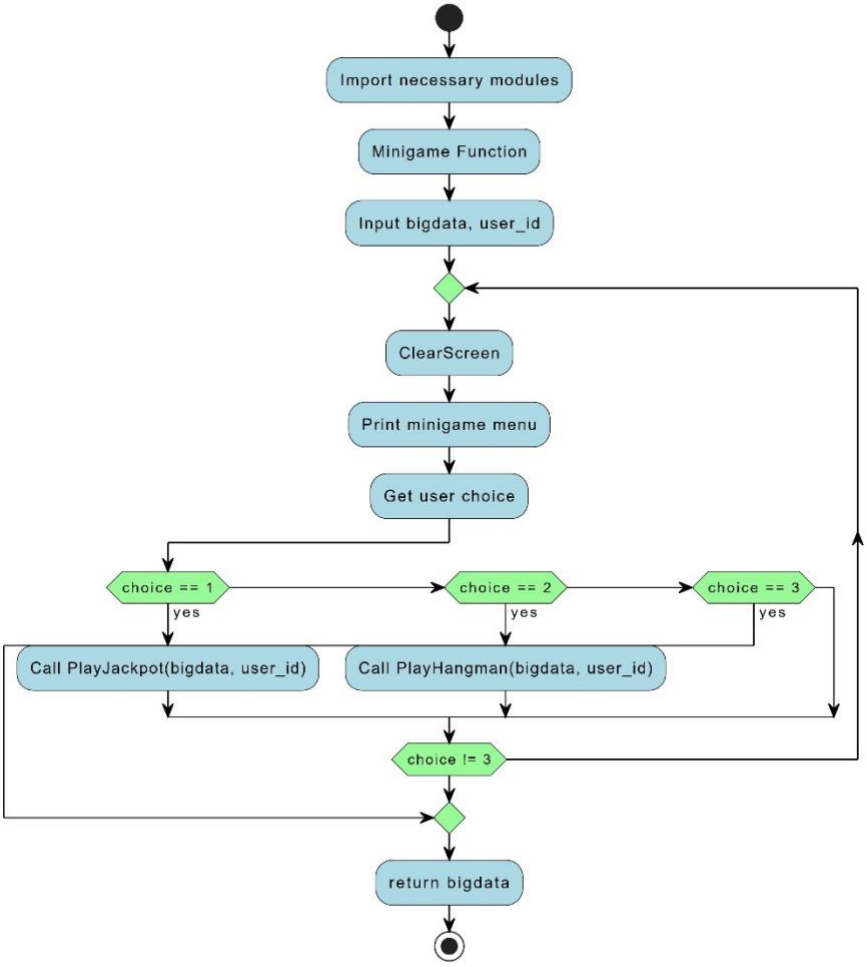
Gambar 19 : A_Functions

B03_MonsterBall



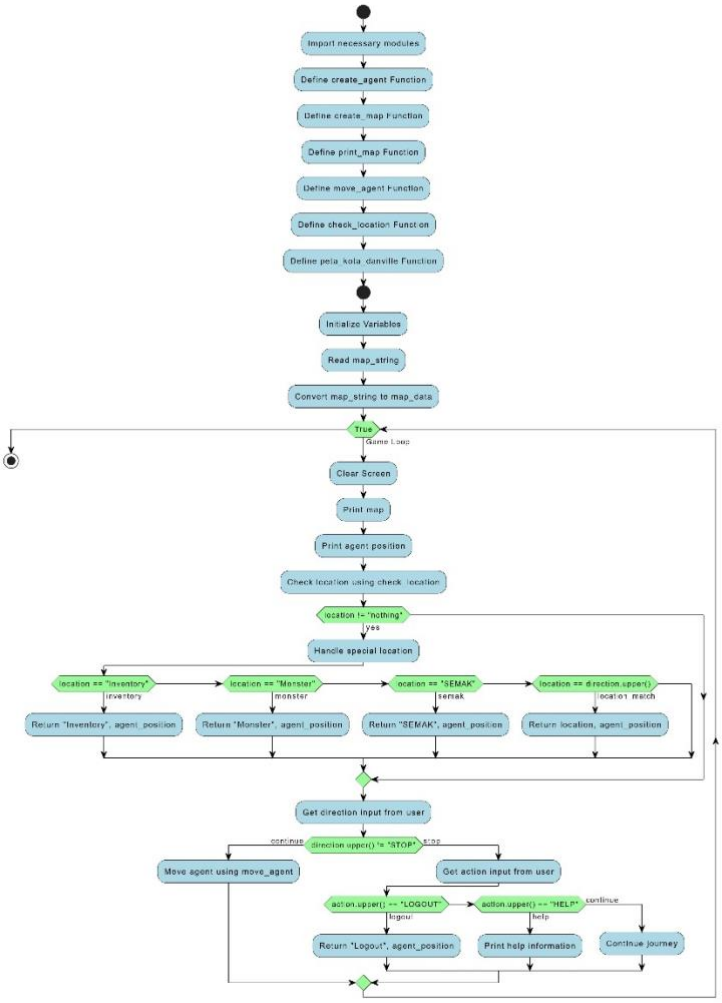
Gambar 20 : B03 – MonsterBall

B04_MiniGame



Gambar 21 : B04 - Minigame

B05_Peta_Kota_Danville



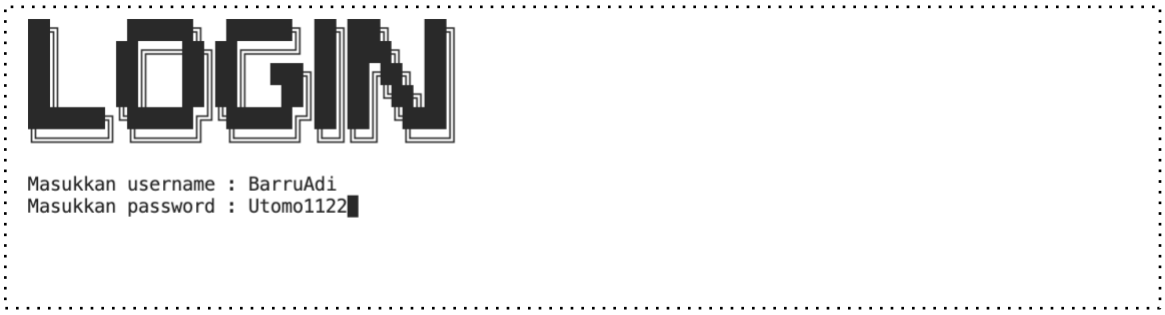
Gambar 22 : B05 - Peta Kota Danville

PENGUJIAN

F01 Register:

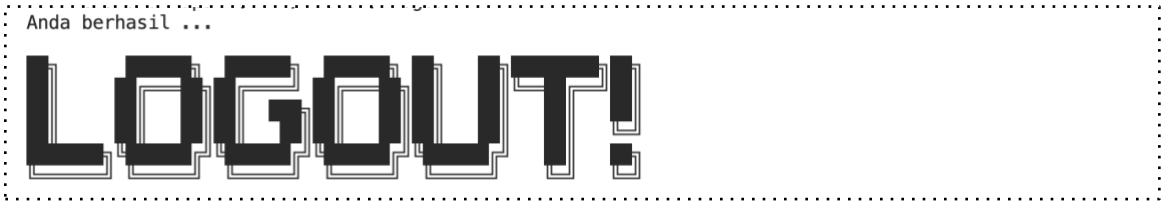
Gambar 21: Pengujian F01_Register

F02 Login



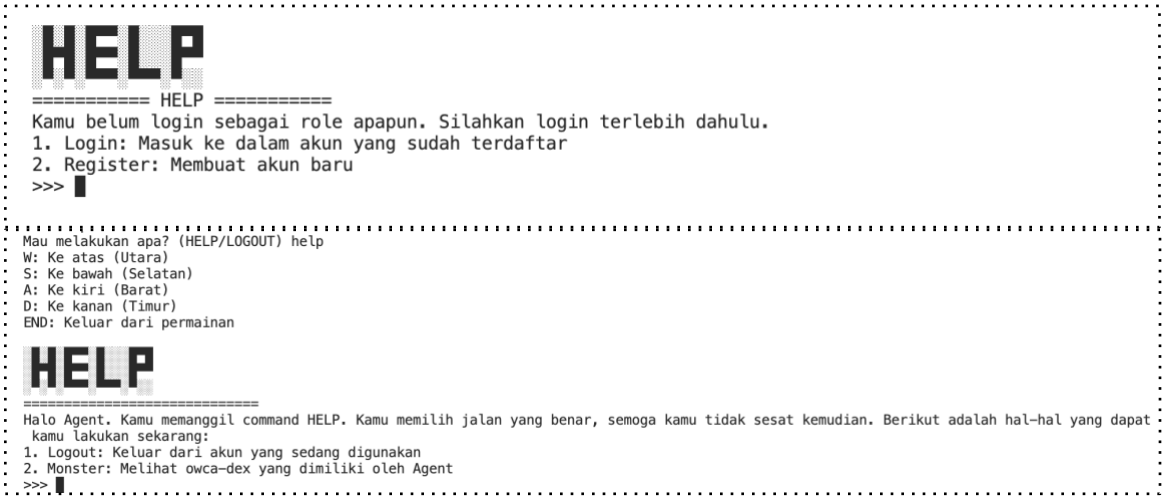
Gambar 22: Pengujian F02_Logout

F03 Logout



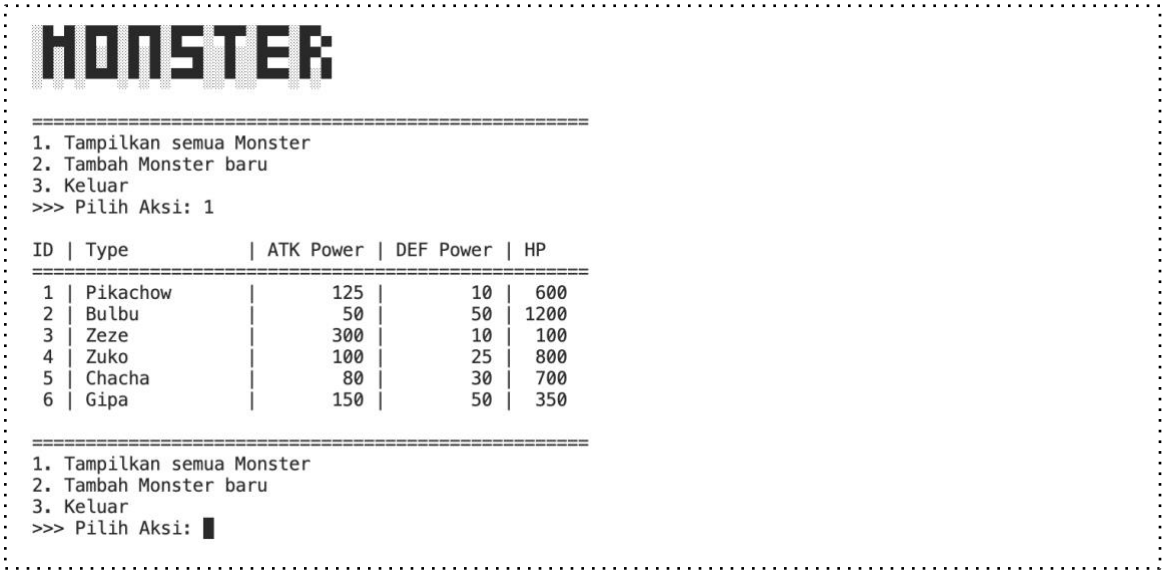
Gambar 23: Pengujian F03_Logout

F04 Help



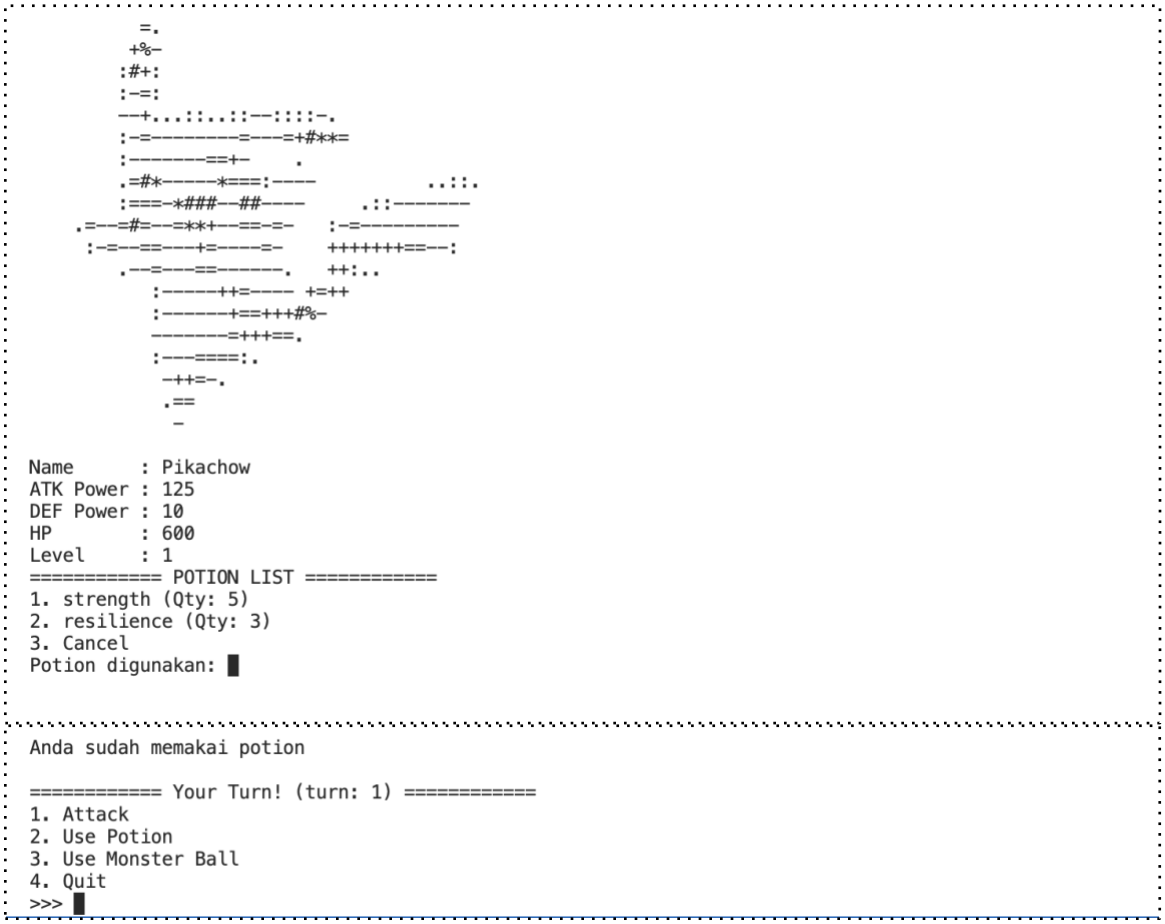
Gambar 24: Pengujian F04_Help

F05 Monster



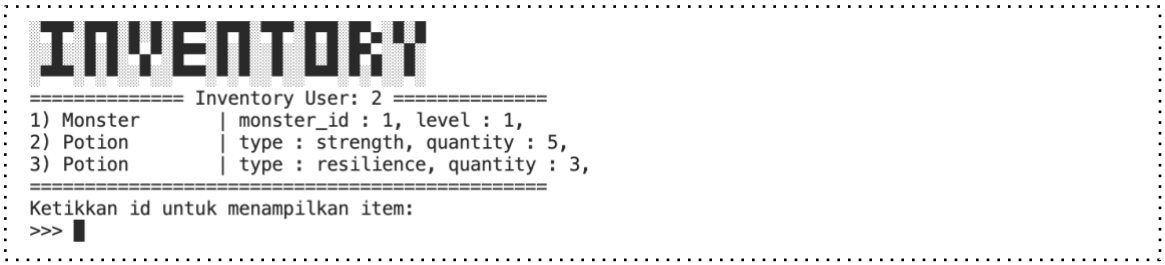
Gambar 25: Pengujian F05_Monster

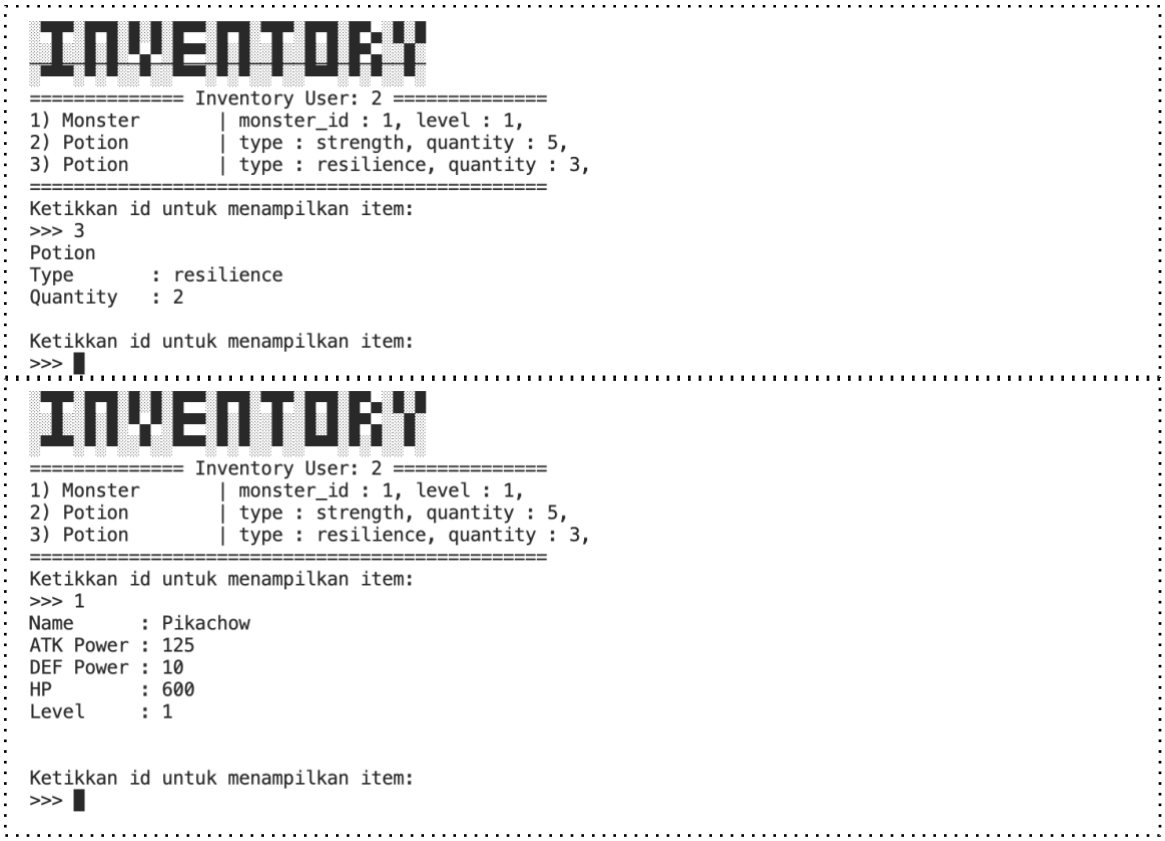
F06 Potion



Gambar 26: Pengujian F06_Potion

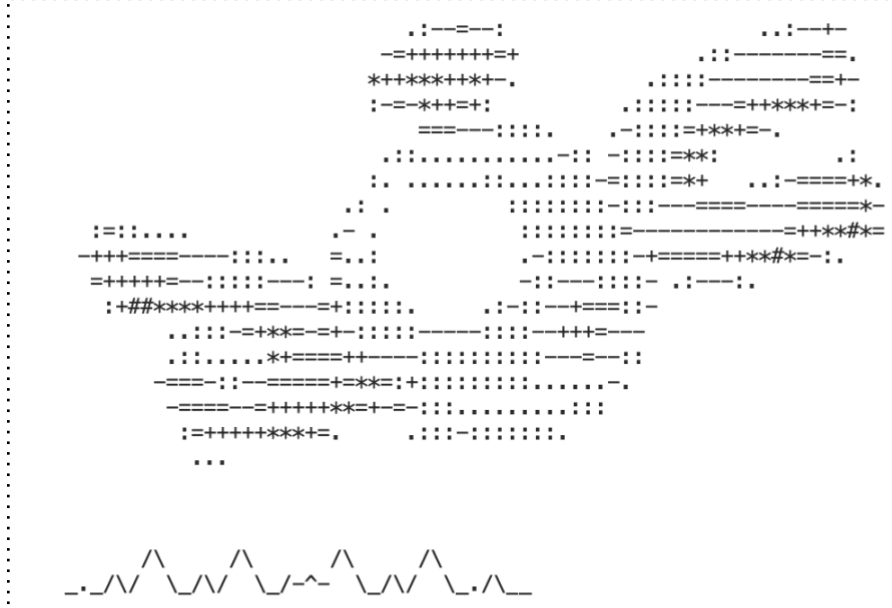
F07 Inventory





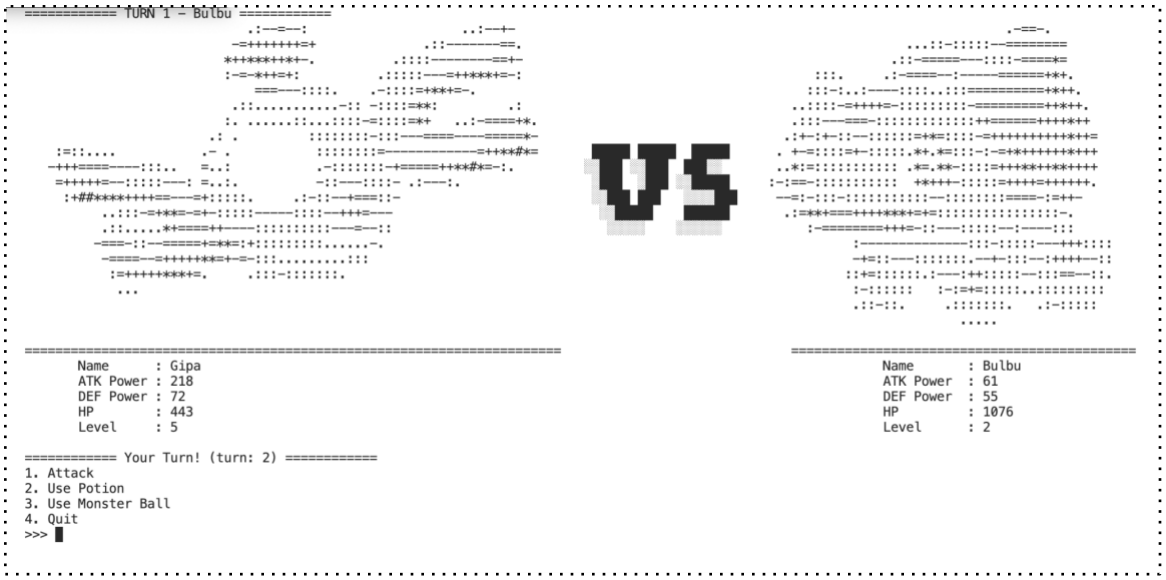
Gambar 27: Pengujian F07_Inventory

F08 Battle



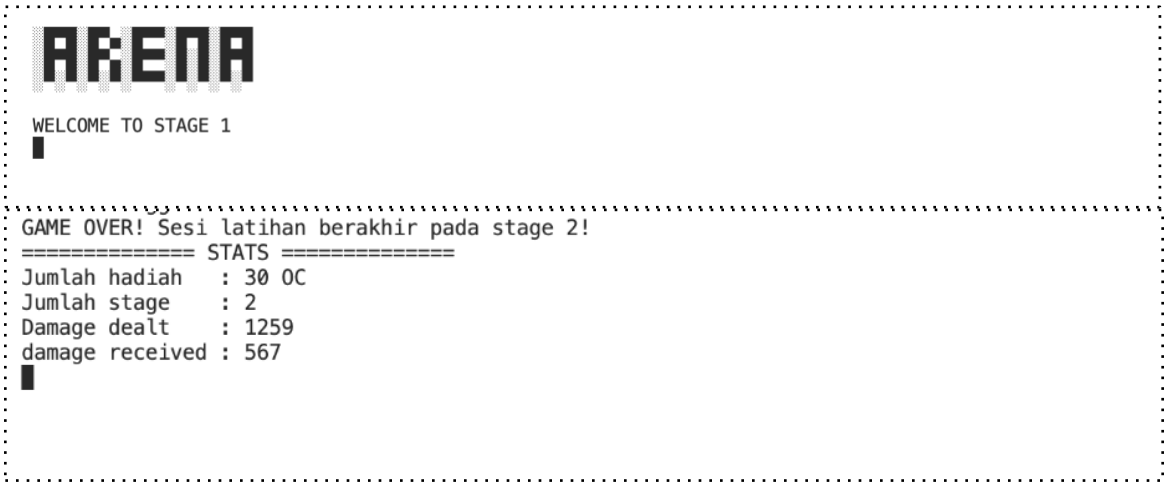
```
RAWRRR, Monster Gipa telah muncul !!!

Name      : Gipa
ATK Power : 218
DEF Power : 72
HP        : 511
Level     : 5
```

```
GO, Bulbu!!

Name      : Bulbu
ATK Power : 55
DEF Power : 55
HP        : 1320
Level     : 2

============ Your Turn! (turn: 1) ============
1. Attack
2. Use Potion
3. Use Monster Ball
4. Quit
>>> 
```



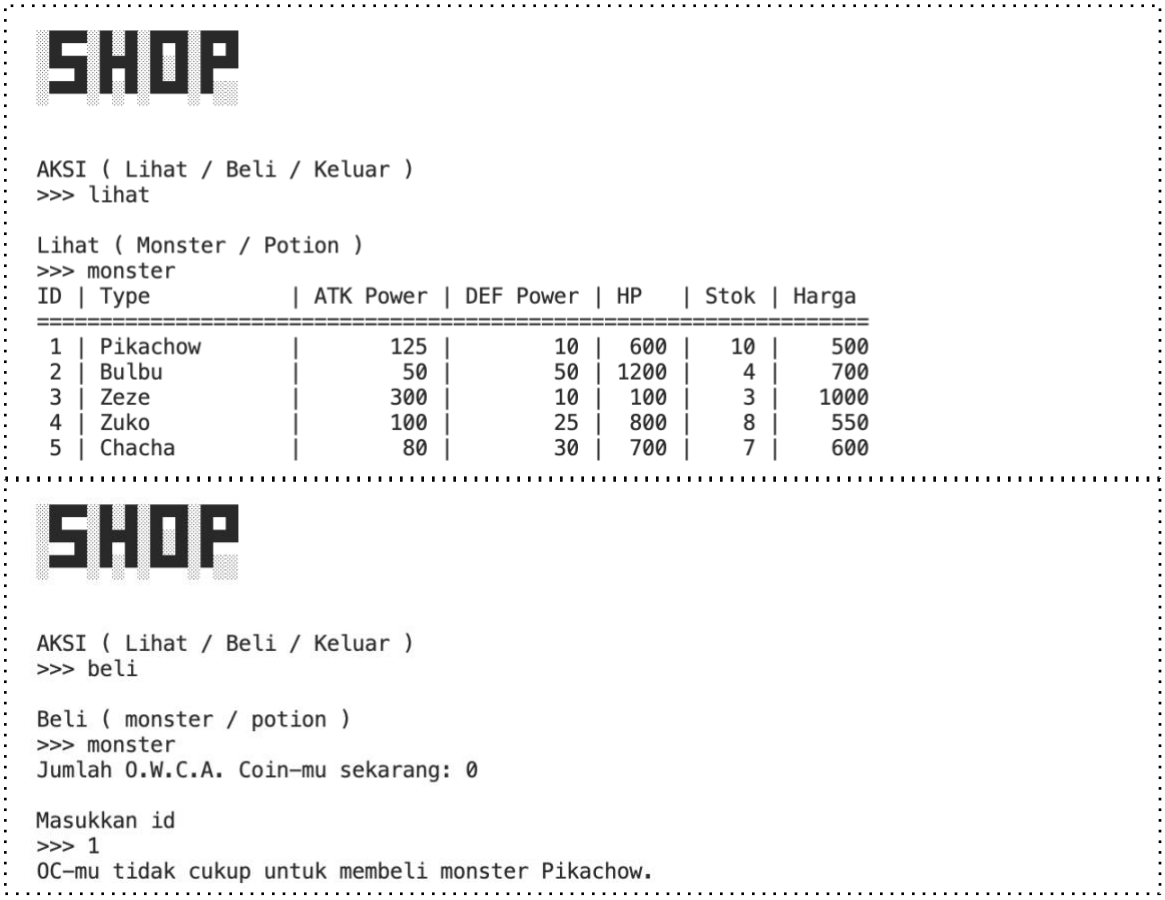
Gambar 28: Pengujian F08_Battle

F09 Arena



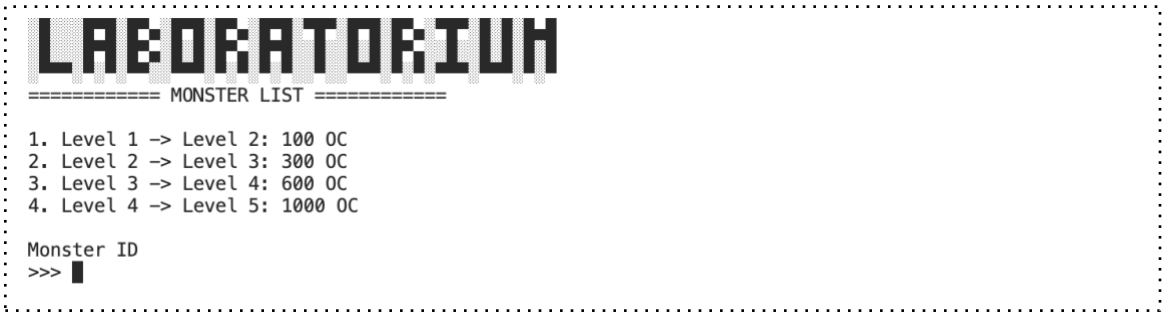
Gambar 29: Pengujian F09_Arena

F10 Shop and Currency



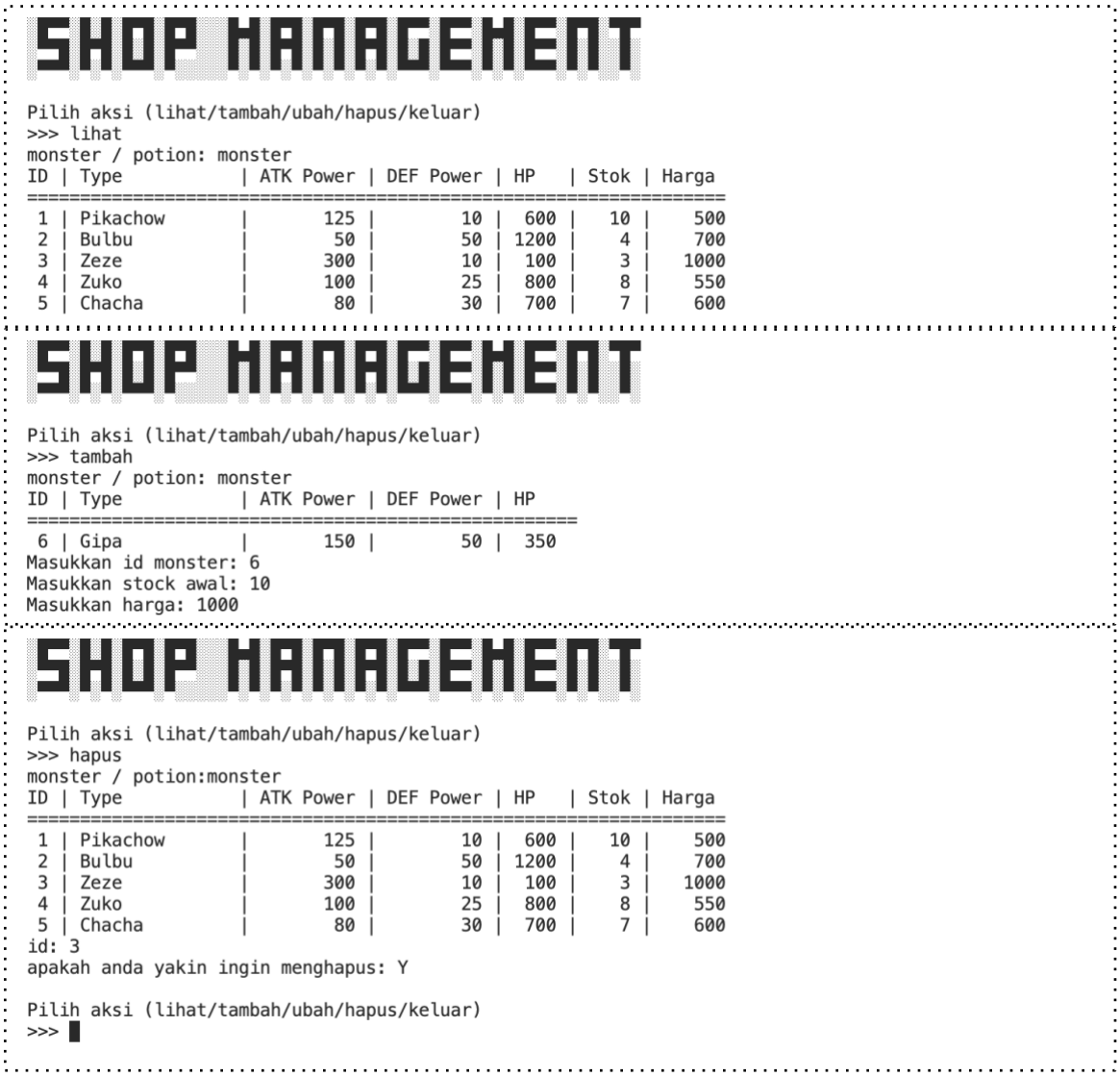
Gambar 30: Pengujian F10_Shop & Currency

F11 Laboratory



Gambar 31: Pengujian F11_Laboratory

F12 Shop Management



Gambar 32: Pengujian F12_Shop Management

F13 Monster Management

```
MONSTER

==================================================
1. Tampilkan semua Monster
2. Tambah Monster baru
3. Keluar
>>> Pilih Aksi: 2
Mulai pembuatan monster!!
Masukkan nama monster: Blublub
>>> Masukkan ATK Power: 300
>>> Masukkan DEF Power: 30
>>> Masukkan HP: 500
Monster baru berhasil dibuat
>>> Tambahkan Monster ke database (Y/N): Y
```

```
>>> Pilih Aksi: 1

ID | Type           | ATK Power | DEF Power | HP
==================================================
 1 | Pikachow       |       125 |        10 |  600
 2 | Bulbu          |        50 |        50 | 1200
 3 | Zeze           |       300 |        10 |  100
 4 | Zuko           |       100 |        25 |  800
 5 | Chacha         |        80 |        30 |  700
 6 | Gipa           |       150 |        50 |  350
 7 | Blublub        |       300 |        30 |  500

==================================================
1. Tampilkan semua Monster
2. Tambah Monster baru
3. Keluar
>>> Pilih Aksi:
```

```
>>> Pilih Aksi: 1

ID | Type           | ATK Power | DEF Power | HP
==================================================
 1 | Pikachow       |       125 |        10 |  600
 2 | Bulbu          |        50 |        50 | 1200
 3 | Zeze           |       300 |        10 |  100
 4 | Zuko           |       100 |        25 |  800
 5 | Chacha         |        80 |        30 |  700
 6 | Gipa           |       150 |        50 |  350
 7 | Blublub        |       300 |        30 |  500

==================================================
1. Tampilkan semua Monster
2. Tambah Monster baru
3. Keluar
>>> Pilih Aksi:
```

Gambar 33: Pengujian F13_Monster Management

F14 Load

```
LOADING...
```

Gambar 34: Pengujian F14_Load

F15 Save

```
Masukkan nama folder: save1

Membuat folder save1...
Data telah disimpan pada folder save1!
```

Gambar 35: Pengujian F15_Save

F16 Exit

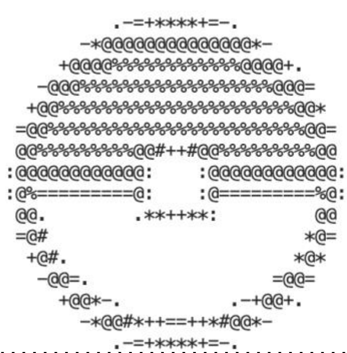
```
EXIT
Apakah anda ingin melakukan penyimpanan?
Y/N: Y
```

```
EXIT
Apakah anda ingin melakukan penyimpanan?
Y/N: N
Terima kasih karena anda telah memainkan game ini! Sampai jumpa! \(\ ( ͡° ͜ʖ ͡°) )/
```

Gambar 36: Pengujian F16_Exit

B03 Monster Ball

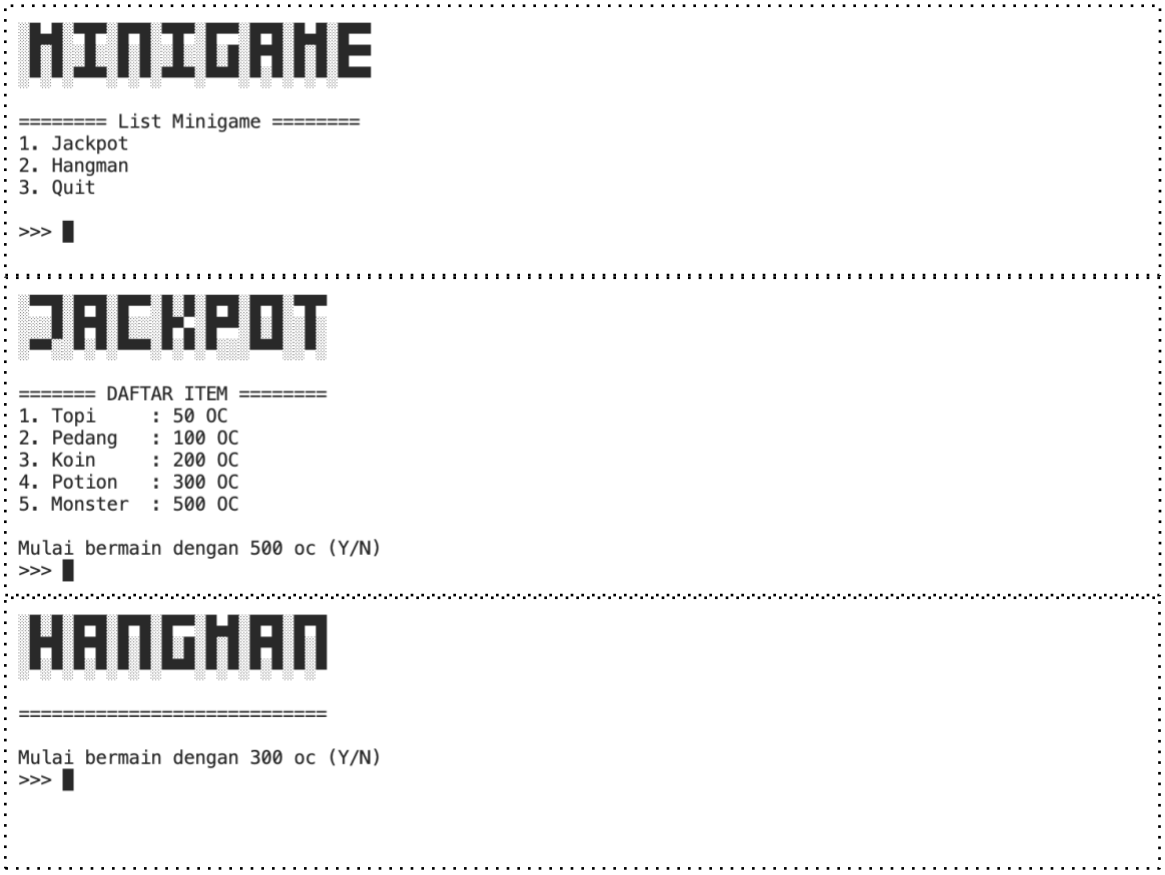
```
Kamu berhasil menangkap
```



```
Anda tidak mempunyai monsterball.

=========== Your Turn! (turn: 1) ===========
1. Attack
2. Use Potion
3. Use Monster Ball
4. Quit
>>>
```

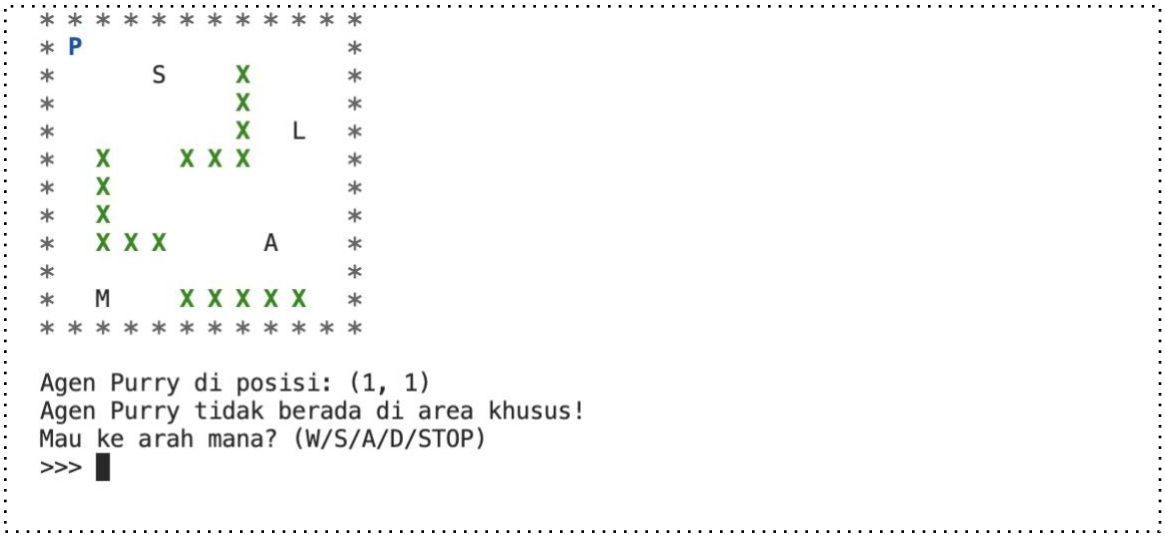
Gambar 37: Pengujian B03_Monster Ball

B04 Minigame



Gambar 38: Pengujian B04_Minigame

B05 Peta Kota Danville



```
* * * * * * * * * * * *
*                     *
*       S     X       *
*             X       *
*             X   L   *
*   X     X X X       *
*   X     P           *
*   X                 *
*   X X X       A     *
*                     *
*   M     X X X X X   *
* * * * * * * * * * * *

Agen Purry di posisi: (6, 5)
Agen Purry akan mengakses Semak, karena berada pada posisi yang bersebelahan dengan X
Mau ke arah mana? (W/S/A/D/STOP)
>>> █
```

```
* * * * * * * * * * * *
*                     *
*       S     X       *
*             X       *
*             X   L   *
*   X     X X X       *
*   X                 *
*   X                 *
*   X X X     P A     *
*                     *
*   M     X X X X X   *
* * * * * * * * * * * *

Agen Purry di posisi: (8, 7)
Agen Purry akan mengakses Arena, karena berada pada posisi yang bersebelahan dengan A
Mau ke arah mana? (W/S/A/D/STOP)
>>> █
```

Gambar 39: Pengujian B05_Peta Kota Danville

SPESIFIKASI PROGRAM

MAIN {Fitur untuk menjalankan inti program}

```
if __name__ == "__main__" then
    main()

procedure main()
    user_data <- {"user_id": None, "username": None, "role": None,
"status": False}

    ClearScreen()
    output("Welcome to the game!")
    loaded, bigdata <- Load()

    if not loaded then
        output("Failed to load the game data.")
    else
        iterate
            choice <- input(">>> ").upper()

            if choice == "REGISTER" then
                ClearScreen()
                data <- BerikanData("user.csv")
                monster_data <- BerikanData("monster.csv")
                Registrasi(data, monster_data)

            else if choice == "LOGIN" then
                if not user_data['status'] then
                    ClearScreen()
                    user_data <- login()
                    agent_position <- (1, 1)

                    if user_data['status'] then
                        loggedin <- True
```

```
                        repeat
                            condition, agent_position <-
peta_kota_danville(agent_position)
                            output(condition)

                            if condition == "Semak" then
                                oc_gain <- Battle(bigdata,
user_id=user_data["user_id"])

                            else if condition == "Arena" then
                                Arena(bigdata,
user_id=user_data['user_id'])

                            else if condition == "Inventory" then
                                ShowInventory(bigdata,
user_id=user_data['user_id'])

                            else if condition == "Shop" and
user_data['role'] == 'agent' then
                                bigdata <- Shop(bigdata,
user_id=user_data['user_id'])

                            else if condition == "Shop" and
user_data['role'] == 'admin' then
                                bigdata <- ShopManagement(bigdata)

                            else if condition == "Laboratorium"
then
                                bigdata <- Laboratory(bigdata,
user_id=user_data['user_id'])

                            else if condition == "Monster" and
user_data['role'] == 'admin' then
                                bigdata <-
MonsterManagement(bigdata)
                        until condition == "End"
                    else
```

```
                        output("Kamu sudah login")

                else
                    output("Kamu sudah login")

            else if choice == "HELP" then
                ClearScreen()
                Help2(user_data)

            else if choice == "LOGOUT" then
                if logout(login_state=user_data["status"]) then
                    user_data <- {"user_id": None, "username":
None, "role": None, "status": False}

            else
                output("Invalid choice, please try again.")
```

F00 RNG {Fitur untuk menghasilkan bilangan asal dalam interval}

```
function RNG(interval: int) -> int
  import datetime as time
  a <- 16523
  c <- 19623
  m <- 10923

{Definisikan seed dengan nilai awal waktu (timestamp)}
  seed <- int(time.datetime.now().timestamp())

{ Menggunakan algoritma LCG untuk menghasilkan bilangan acak dalam
rentang interval}
  random_number <- (((a * seed + c) % m) % interval) + 1

  -> random_number
```

F01 - Register {Fitur untuk proses Registrasi}

```
function Registrasi(bigdata: dict)
    output()
    output(text_ascii['register'])
    output()

    // Mengubah data pengguna dan data monster menjadi list of
lists
    data ← [[value repeat value in user.values()] repeat user in
bigdata['user']]
    monster_data ← [[value repeat value in user.values()] repeat
user in bigdata['monster']]

    iterate
        // Meminta input username dan password dari pengguna
        usernamePendaftar ← input("Masukkan username : ")
        passwordPendaftar ← input("Masukkan password : ")

        // Memeriksa apakah username sudah digunakan
        if PeriksaUsernameUnik(data, usernamePendaftar) == False
then
            // Jika username sudah digunakan, tampilkan pesan dan
ulangi input
            output("Username", usernamePendaftar, "sudah terpakai!
Silakan isi ulang form dengan username lain.")
            output()
        else
            // Jika username belum digunakan, periksa ketentuan
password
            PeriksaKPassword(bigdata, usernamePendaftar,
passwordPendaftar, monster_data, data)
            output(bigdata['user'])
```

```
            -> bigdata
    until break


function PeriksaUsernameUnik(data: list of lists,
usernameApplicant: string) -> boolean
    repeat each row in data
        if length of row >= 2 and usernameApplicant = row[1] then
            -> False
    -> True


function PeriksaKUsername(usernamePendaftar: string,
passwordPendaftar: string, monster_data: list of lists, data: list
of lists) -> void
    is_valid <- True
    repeat each char in usernamePendaftar
        if not (char is alphanumeric or char = "_" or char = "-")
then
            is_valid <- False
    if is_valid then
        TambahAkunBaru(usernamePendaftar, passwordPendaftar,
monster_data, data)
    else
        output("Username hanya dapat mengandung huruf alfabet (A-
Za-z), underscore (_), strip (-), dan angka (0-9). Silakan isi
ulang form dengan username yang sesuai.")
        Registrasi(data, monster_data)


function PeriksaKPassword(usernamePendaftar: string,
passwordPendaftar: string, monster_data: list of lists, data: list
of lists) -> void
    if length(passwordPendaftar) >= 8 then
        has_upper <- False
        has_lower <- False
        has_digit <- False
```

```
        repeat each char in passwordPendaftar do
            if char is uppercase then
                has_upper <- True
            else if char is lowercase then
                has_lower <- True
            else if char is digit then
                has_digit <- True
        if has_upper and has_lower and has_digit then
            if usernamePendaftar not in passwordPendaftar then
                is_valid <- True
                repeat each char in passwordPendaftar do
                    if not (char is alphanumeric or char = "_" or
char = "-") then
                        is_valid <- False
                if is_valid then
                    if ' ' not in passwordPendaftar then
                        PeriksaKUsername(usernamePendaftar,
passwordPendaftar, monster_data, data)
                    else
                        output("Password tidak boleh mengandung
spasi. Silakan isi ulang form dengan password yang sesuai.")
                        Registrasi(data, monster_data)
                else
                    output("Password hanya dapat mengandung huruf
alfabet (A-Za-z), underscore (_), strip (-), dan angka (0-9).
Silakan isi ulang form dengan password yang sesuai.")
                    Registrasi(data, monster_data)
            else
                output("Password tidak boleh mengandung username.
Silakan isi ulang form dengan password yang sesuai.")
                Registrasi(data, monster_data)
        else
            output("Password harus terdiri dari setidaknya satu
huruf besar (A-Z), satu huruf kecil (a-z), dan satu angka (0-9).
Silakan isi ulang form dengan password yang sesuai.")
            Registrasi(data, monster_data)
    else
```

```
        output("Password harus memiliki panjang minimal 8
karakter. Silakan isi ulang form dengan password yang sesuai.")
        Registrasi(data, monster_data)



function PilihMonster(monster_data)
    iterate
        output("Silahkan pilih salah satu monster sebagai monster
awalmu.")
        output("1. Pikachow")
        output("2. Bulbu")
        output("3. Zeze")
        output("4. Zuko")
        output("5. Chacha")

        pilihan <- ""

        iterate
            pilihan <- input("Monster Pilihanmu: ")
            if pilihan != "1" and pilihan != "2" and pilihan !=
"3" and pilihan != "4" and pilihan != "5" then
                output("Ulangi Input!")
        until pilihan == "1" or pilihan == "2" or pilihan == "3"
or pilihan == "4" or pilihan == "5"

        b <- int(pilihan)

        if pilihan.isdigit() and 1 <= int(pilihan) <=
len(monster_data) - 1 then
            monster_id <- int(pilihan)
            monster_type <- monster_data[monster_id][1]
            monster_atk_power <- monster_data[monster_id][2]
            monster_def_power <- monster_data[monster_id][3]
            monster_hp <- monster_data[monster_id][4]

            if b == 1 then
```

```
                output(monster_ascii['Pikachow'])
            else if b == 2 then
                output(monster_ascii['Bulbu'])
            else if b == 3 then
                output(monster_ascii['Zeze'])
            else if b == 4 then
                output(monster_ascii['Zuko'])
            else if b == 5 then
                output(monster_ascii['Chacha'])

            output("Anda memilih monster dengan detail sebagai
berikut:")
            output("Type: " + monster_type)
            output("ATK Power: " + str(monster_atk_power))
            output("DEF Power: " + str(monster_def_power))
            output("HP: " + str(monster_hp))

            a <- input("Kamu yakin memilih monster ini? (y/n) ")

            if a == "y" then
                -> int(pilihan)
            else if a == "n" then
                output("Silahkan Pilih lagi!")
                output()
            else
                output("Invalid input, pilih lagi!!")
        else
            output("ID monster tidak valid. Silakan pilih lagi!")
    until a == "y"

// TambahAkunBaru
procedure TambahAkunBaru(bigdata: dict, usernamePendaftar: str,
passwordPendaftar: str, monster_data: dict, data: list) -> dict:
    // Menemukan ID terakhir dan menambahkannya satu angka di
atasnya
    id_akhir <- data[-1]
    last_id <- int(data[-1][0])
```

```
    new_id <- last_id + 1

    // Memilih monster
    monster_id <- PilihMonster(monster_data)
    monster_type <- monster_data[monster_id][1]

    // Menampilkan informasi registrasi
    output("Registrasi berhasil untuk username",
usernamePendaftar, "dengan monster", monster_type)
    data_user <- {
        'id': new_id,
        'username': usernamePendaftar,
        'password': passwordPendaftar,
        'role': 'agent',
        'oc': 0,
    }
    data_monster <- {
       'user_id': new_id,
       'monster_id': monster_id,
       'level': 1,
    }

    // Menyimpan perubahan ke file CSV
    append data_monster to bigdata['monster_inventory']
    append data_user to bigdata['user']
    return bigdata
```

F02 - Login {Fitur untuk melakukan proses login}

```
function get_user_role(username: string) -> string or None
    // Memuat data pengguna dari file CSV
    user_data <- BerikanData("user.csv")
    // Inisialisasi peran pengguna
    role <- None
    // Mencari peran pengguna berdasarkan username
```

```
    i traversal [1..length(user_data)]
        user <- user_data[i]
        if length(user) >= 4 then  // Pastikan ada cukup banyak
kolom dalam baris pengguna
            if username == user[1] then
                role <- user[3]  // Mengembalikan peran pengguna
                -> role
    -> role  // Mengembalikan None jika username tidak ditemukan

function get_user_id(username: string) -> int or None
    // Memuat data pengguna dari file CSV
    user_data <- BerikanData("user.csv")
    // Inisialisasi id pengguna
    user_id <- None
    // Mencari id pengguna berdasarkan username
    i traversal [1..length(user_data)]
        user <- user_data[i]
        if length(user) >= 4 then  // Pastikan ada cukup banyak
kolom dalam baris pengguna
            if username == user[1] then
                user_id <- integer(user[0])  // Mengembalikan id
pengguna
                -> user_id
    -> user_id  // Mengembalikan None jika username tidak
ditemukan
// Ingat kalau user_id = None, maka return user_id akan memberikan
None

function enkripsi_password(password: string) -> string
    // Inisialisasi string kosong untuk password yang dienkripsi
    encrypted_password <- ""
    // Alfabet kecil dan besar sebagai referensi
    alfabet_kecil <- 'abcdefghijklmnopqrstuvwxyz'
    alfabet_besar <- 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    // Enkripsi setiap karakter dalam password
    repeat length(password) times
        character <- password[index]
```

```
        if character.islower() then
            posisi <- alfabet_kecil.find(character)
            posisi_baru <- (posisi + 3) % 26
            encrypted_password <- encrypted_password +
alfabet_kecil[posisi_baru]
        else if character.isupper() then
            posisi <- alfabet_besar.find(character)
            posisi_baru <- (posisi + 3) % 26
            encrypted_password <- encrypted_password +
alfabet_besar[posisi_baru]
        else
            encrypted_password <- encrypted_password + character
    -> encrypted_password

function dekripsi_password(encrypted_password: string) -> string
    // Inisialisasi string kosong untuk password yang didekripsi
    decrypted_password <- ""
    // Alfabet kecil dan besar sebagai referensi
    alfabet_kecil <- 'abcdefghijklmnopqrstuvwxyz'
    alfabet_besar <- 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    // Dekripsi setiap karakter dalam password yang dienkripsi
    repeat length(encrypted_password) times
        character <- encrypted_password[index]
        if character.islower() then
            posisi <- alfabet_kecil.find(character)
            posisi_baru <- (posisi - 3) % 26
            decrypted_password <- decrypted_password +
alfabet_kecil[posisi_baru]
        else if character.isupper() then
            posisi <- alfabet_besar.find(character)
            posisi_baru <- (posisi - 3) % 26
            decrypted_password <- decrypted_password +
alfabet_besar[posisi_baru]
        else
            decrypted_password <- decrypted_password + character
    -> decrypted_password
```

```
function login() -> dictionary
    // Output teks login
    Output text_ascii['login']
    // Inisialisasi data pengguna
    user_data <- {"user_id": None, "username": None, "role": None,
"status": False}
    // Masukkan username dan password
    usernamePengguna <- input("Masukkan username: ")
    passwordPengguna <- input("Masukkan password: ")
    // Memuat data pengguna dari file CSV
    matriks_user <- BerikanData("user.csv")
    // Inisialisasi status cek username dan password
    check_username <- False
    check_pw <- False
    // Loop melalui data pengguna untuk memeriksa kecocokan
username dan password
    repeat length(matriks_user) times
        user <- matriks_user[index]
        if length(user) >= 3 then  // Pastikan ada cukup banyak
kolom dalam baris data yang diberikan
            if usernamePengguna == user[1] then
                check_username <- True
                if passwordPengguna == user[2] then
                    check_pw <- True
                    -> Exit repeat  // Keluar dari loop jika
password cocok ditemukan
    // Cek hasil pencarian username dan password
    if check_username then
        if check_pw then
            // Set data pengguna setelah login berhasil
            user_data['user_id'] <-
integer(get_user_id(usernamePengguna))
            user_data['username'] <- usernamePengguna
            user_data['role'] <- get_user_role(usernamePengguna)
            user_data['status'] <- True
            Output "Login berhasil!"
        else
```

```
            Output "Password salah"
    else
        Output "Username salah"
    -> user_data
```

F03 - Logout {Fitur untuk proses logout}

```
function logout(login_state: bool) -> bool
  # Lakukan login terlebih dahulu
  if login_state == True then
      Output("Anda berhasil ...")
      Output(text_ascii['logout'])
      -> True
  else
      Output("Anda tidak berhasil logout.")
      -> False
```

F04 - Help {Fitur untuk membantu pengguna program}

```
function Help2(status : Dict) -> string
    if not status['status'] then
        output("=========== HELP ===========")
        output("Kamu belum login sebagai role apapun. Silahkan
login terlebih dahulu.")
        output("1. Login: Masuk ke dalam akun yang sudah
terdaftar")
        output("2. Register: Membuat akun baru")
        choice ← input(">>> ")
        -> choice
    else
        // Memeriksa apakah pengguna adalah admin
```

```
        if status['role'] == 'admin' then
            output("========== HELP ==========")
            output("Halo Admin. Kamu memanggil command HELP. Kamu
memilih jalan yang benar, semoga kamu tidak sesat kemudian.
Berikut adalah hal-hal yang dapat kamu lakukan sekarang:")
            output("1. Logout: Keluar dari akun yang sedang
digunakan")
            output("2. Monster: Mengatur Monster")
            output("3. Shop: Melakukan manajemen pada SHOP sebagai
tempat jual beli peralatan Agent")
            choice ← input(">>> ")
            -> choice
        else
            // Asumsi pengguna adalah agent
            output("========== HELP ==========")
            output("Halo Agent. Kamu memanggil command HELP. Kamu
memilih jalan yang benar, semoga kamu tidak sesat kemudian.
Berikut adalah hal-hal yang dapat kamu lakukan sekarang:")
            output("1. Logout: Keluar dari akun yang sedang
digunakan")
            output("2. Monster: Melihat owca-dex yang dimiliki
oleh Agent")
            choice ← input(">>> ")
            -> choice
        end if
    end if
end function
```

F05 - Monster {Fitur untuk mempersiapkan monster}

```
function MonsterLevel(monster_data: dict, n_times: int) -> dict
    repeat (n_times - 1) times do
```

```
        attack: int <- int(monster_data['atk_power'] * 11/10)
        defense: int <- int(monster_data['def_power'] * 11/10)
        hp: int <- int(monster_data['hp'] * 11/10)
        monster_data["atk_power"] <- attack
        monster_data["def_power"] <- defense
        monster_data["hp"] <- hp
    -> monster_data

function Attack(monster_atk: int, monster_def: int) -> int
    damage <- monster_atk * (130 - RNG(60)) / 100
    damage -= damage * Defend(monster_def)
    -> int(damage)

function Defend(monster_def: int) -> float
    damage_reduction <- RNG(monster_def) / 100
    -> damage_reduction

function MonsterDetail(bigdata, monster_id: int, level: int = 1) -
> dict
    monster_data <- bigdata['monster'][1:]
    repeat data in monster_data
        if int(data['id']) == int(monster_id) then
            data <- MonsterLevel(data, level)
            -> data
```

F06 - Potion {Fitur untuk menyusun identitas potion}

```
function PotionMenu(user_inventory: list, monster_data: dict) ->
(dict, bool)
    count <- 1
```

```
    for i <- 0 to length of user_inventory - 1 do
        output ({count} + {user_inventory[i]['type']} + (Qty:
{user_inventory[i]['quantity']})
        count <- count + 1
    output ({count} +. Cancel)

    user_choice <- integer value of input("Potion digunakan: ")
    if user_choice <= length of user_inventory then
        user_inventory[user_choice - 1]['quantity'] <-
user_inventory[user_choice - 1]['quantity'] - 1
        if user_inventory[user_choice - 1]['type'] == 'strength'
then
            monster_data <- StrengthPotion(monster_data)
        else if user_inventory[user_choice - 1]['type'] ==
'resilience' then
            monster_data <- ResiliencePotion(monster_data)
        else if user_inventory[user_choice - 1]['type'] ==
'healing' then
            monster_data <- HealingPotion(monster_data)
        ClearScreen()
        -> monster_data, true
    else
         if user_choice == (length of user_inventory + 1) then
            ClearScreen()
            -> monster_data, false
         else
            output("Invalid choice.")
            -> monster_data, false

function StrengthPotion(monster_data: dict) -> dict
    monster_data['atk_power'] <-
round(float(monster_data['atk_power'] * 1.05))
    -> monster_data

function ResiliencePotion(monster_data: dict) -> dict
    monster_data['def_power'] <-
round(float(monster_data['def_power'] * 1.05))
```

```
    -> monster_data

function HealingPotion(current_monster_data: dict, monster_data:
dict) -> dict
    current_monster_data['hp'] <- current_monster_data['hp'] *
1.25
    if current_monster_data['hp'] > monster_data['hp'] then
        current_monster_data['hp'] <- monster_data['hp']
    -> current_monster_data
```

F07 - Inventory {Fitur untuk menyusun identitas Inventory}

```
procedure displayMonsterStat(monster_data: dict, monster_level:
int) -> none:
    output("Name      :", monster_data["type"])
    output("ATK Power :", monster_data["atk_power"])
    output("DEF Power :", monster_data["def_power"])
    output("HP        :", monster_data["hp"])
    output("Level     :", monster_level)

procedure showInventory(bigdata, user_id) -> none:
    inventory <- call inventoryUserDict(bigdata, user_id <-
user_id)
    data_monster <- inventory['monster']
    data_item <- inventory['item']
    counter <- 1
    inventory_2 <- []

    repeat (length(data_monster)) times
        temp <- ['monster']
        repeat values in inventory['monster'][repeat_index - 1]
            if values != 'user_id' then
                add values to temp
        add temp to inventory_2
```

```
    repeat (length(data_item)) times
        temp <- ['item']
        repeat each values in inventory['item'][repeat_index - 1]
            if values != 'user_id' then
                add values to temp
        add temp to inventory_2

    output("============== Inventory User:", user_id,
"==============")
    banyak_monster <- length(inventory['monster'])
    repeat (banyak_monster) times
        output(counter, ")", "Monster       | ")
        repeat each key, values in
inventory['monster'][repeat_index - 1]
            if key != 'user_id' then
                output(key, ':', values)
        output
        counter <- counter + 1

    banyak_item <- length(inventory['item'])
    repeat (banyak_item) times
        output(counter, ")",
        if inventory['item'][repeat_index - 1]['type'] ==
'monsterball' then
            output("Monsterball   | ")
        else
            output("Potion        | ")
        repeat key, values in inventory['item'][repeat_index - 1]
            if key != 'user_id' then
                output(key, ':', values)
        output()
        counter <- counter + 1

    output("================================================")
    output("Ketikkan id untuk menampilkan item:")
    id <- input().upper()
```

```
    iterate id != "KELUAR"
        if id.isdigit() then
            call detailItem(bigdata, id <- convertToInt(id),
inventory_2)
        output("Ketikkan id untuk menampilkan item:")
        id <- input().upper()
    until id == "KELUAR"


procedure detailItem(bigdata, item_id, inventory) -> none:
    banyak_id <- length(inventory)
    if 0 < item_id <= banyak_id then
        if inventory[item_id - 1][0] == 'monster' then
            monster_data <- call monsterDetail(bigdata, monster_id
<- inventory[item_id - 1][2], level <- inventory[item_id - 1][3])
            call displayMonsterStat(monster_data <- monster_data,
monster_level <- inventory[item_id - 1][2])
            output("\n")
        else if inventory[item_id - 1][0] == 'item' then
            if inventory[item_id - 1][2] == 'monsterball' then
                output("MonsterBall")
                output("Quantity:  ", inventory[item_id - 1][1])
                output("\n")
            else
                output("Potion")
                output("Type       :", inventory[item_id - 1][2])
                output("Quantity   :", inventory[item_id - 1][1])
                output("\n")
        else
            output("ID tidak tersedia")

function inventoryUserDict(bigdata: dict, user_id: int) -> dict:
    data_monster <- bigdata['monster_inventory']
    data_item <- bigdata['item_inventory']
    inventory <- {'monster': [], 'item': []}
    repeat i <- 1 to length(data_monster) - 1 times
        if convertToInt(data_monster[i]['user_id']) == user_id
```

```
then
            add data_monster[i] to inventory['monster']

    i transversal
        if convertToInt(data_item[i]['user_id']) == user_id then
            add data_item[i] to inventory['item']
    -> inventory

function userInventory(user_id: int, bigdata: dict, type: str) ->
list:
    data <- bigdata[type][1:]
    list_item <- []
    for each i in data do
        if convertToInt(i['user_id']) == user_id then
            temp <- i
            add temp to list_item
    -> list_item
```

F08 - Battle {Fitur untuk proses battle}

```
procedure printMonster(side: str, monster_data: dict,
monster_level: int, status: str = none) -> none:
    if side == "opponent" then
        output(monster_ascii[monster_data['type']])
        printTerrain('grass')
        if status == "beginning" then
            output(f"RAWRRR, Monster {monster_data['type']} telah
muncul !!!\n")
        displayMonsterStat(monster_data <- monster_data,
monster_level <- monster_level)
    else if side == "player" then
        output(monster_ascii[monster_data['type']])
```

```
        // Print Terrain pokeball
        if status == "beginning" then
            output(f"GO, {monster_data['type']}!!\n")
        displayMonsterStat(monster_data <- monster_data,
monster_level <- monster_level)
    else

procedure displayMonsterStat(monster_data: dict, monster_level:
int) -> none:
    output("Name      :", monster_data["type"])
    output("ATK Power :", monster_data["atk_power"])
    output("DEF Power :", monster_data["def_power"])
    output("HP        :", monster_data["hp"])
    output("Level     :", monster_level)

function monsterAppear(monster_pool: dict):

    monster_appear_id    <- rng(length(monster_pool))
    monster_appear_level <- rng(5)

    -> monster_appear_id, monster_appear_level

function battle(bigdata: dict, user_id: int, in_arena: bool =
false, level_setting: int = none) -> int:
    // Initialize
    clearScreen()
    monster_pool <- ambilData("monster.csv")[1:]
    exit_battle: bool    <- false
    damage_dealt: int    <- 0
    damage_received: int <- 0

    monster_appear_id, monster_appear_level <-
monsterAppear(monster_pool)
    monster_appear <- monster_pool[monster_appear_id - 1]

    if in_arena then
        monster_appear_level <- level_setting
```

```
    monsterLevel(monster_appear, monster_appear_level)

    printMonster(side="opponent", monster_data=monster_appear,
monster_level=monster_appear_level, status="beginning")

    // Monster User
    output("\n============ MONSTER LIST ============")
    user_monster_pool <- userInventory(user_id <- user_id, bigdata
<- bigdata, type <- "monster_inventory")
    user_inventory_pool <- userInventory(user_id <- user_id,
bigdata <- bigdata, type <- "item_inventory")

    // Displaying Monster User List
    id_count <- 1
    repeat monster in user_monster_pool
        output(id_count, monsterDetail(bigdata,
monster['monster_id'], monster['level'])['type'])
        id_count <- id_count + 1

    // Selecting Monster User
    iterate
        user_choice <- input("\nPilih monster untuk bertarung: ")
         if user_choice.isdigit() then
             user_choice <- convertToInt(user_choice)
             if user_choice > length(user_monster_pool) then
                 output("Pilihan nomor tidak tersedia!")
             else
                 user_choice <- user_choice - 1
                 clearScreen()
                 break
     until user_choice.isdigit() and user_choice <=
length(user_monster_pool)


    user_monster_level <- user_monster_pool[user_choice]['level']
    user_monster <- monsterDetail(bigdata, monster_id <-
```

```
user_monster_pool[user_choice]['monster_id'], level <-
user_monster_level)
    // user_monster <- monsterLevel(user_monster, n_times <-
user_monster_level)

    // Printing ASCII art
    printMonster(side="player", monster_data=user_monster,
monster_level=user_monster_level, status='beginning')

    used_potion <- false
    turn_count <- 1

    /
    // Battle Loop
    iterate until (user_monster['hp'] <= 0) or
(monster_appear['hp'] <= 0) or exit_battle
        output(f"\n============ Your Turn! (turn: {turn_count})
============")
        if not in_arena then
            output("""1. Attack\n2. Use Potion\n3. Use Monster
Ball\n4. Quit""")
        else if in_arena then
            output("""1. Attack\n2. Use Potion\n3. Quit""")
        choice <- input(">>> ")


        clearScreen()
        if monster_appear['hp'] < 0 or user_monster['hp'] < 0 then
            break
        else

            if choice == "1" then
                output(f"============ TURN {turn_count} -
{user_monster['type']} ============")

            // User Attacking
            damage_dealt <- damage_dealt +
```

```
attack(user_monster['atk_power'], monster_appear['def_power'])
            monster_appear['hp'] <- monster_appear['hp'] -
attack(user_monster['atk_power'], monster_appear['def_power'])
            // Checking Opponent HP
            if monster_appear['hp'] < 0 then
                monster_appear['hp'] <- 0
                printMonster(side="opponent",
monster_data=monster_appear, monster_level=monster_appear_level)

                break

            // Opponent Attacking
            damage_received <- damage_received +
attack(monster_appear['atk_power'], user_monster['def_power'])
            user_monster['hp'] <- user_monster['hp'] -
attack(monster_appear['atk_power'], user_monster['def_power'])
            if user_monster['hp'] < 0 then
                clearScreen()
                output("Kamu kalah pertandingan ini")
                break

            printMonster(side="opponent",
monster_data=monster_appear, monster_level=monster_appear_level)

            // Processing Other Turn
            time.sleep(rng(2))
            output("\n",text_ascii['vs'])


            printMonster(side="player", monster_data=user_monster,
monster_level=user_monster_level)
            turn_count <- turn_count + 1
            output("\n\n")

        else if choice == "2" and not used_potion then
            output("============ POTION LIST ============")
            user_monster, used_potion <-
```

```
potionMenu(user_inventory_pool, user_monster)
            printMonster(side="player", monster_data=user_monster,
monster_level=user_monster_level)
            continue

        else if choice == "2" and used_potion then
            printMonster(side="player", monster_data=user_monster,
monster_level=user_monster_level)
            output("\nAnda sudah memakai potion")

        else if choice == "3" and not in_arena then
            monster_before <- length(user_monster_pool)

            monsterBall(user_monster_pool, user_id,
monster_appear_id, monster_appear_level)

            monster_after <- length(user_monster_pool)

            if monster_after > monster_before then
                break
            else
                printMonster(side="player",
monster_data=user_monster, monster_level=user_monster_level)

        else if (choice == "4" and not in_arena) or (choice =="3"
and in_arena) then
            exit_battle <- true
            break

        else
            output(f"GO, {user_monster['type']}!!\n")
            printMonster(side="player", monster_data=user_monster,
monster_level=user_monster_level)
            output("\nInput Anda salah!")

    win_battle <- false
    oc_gain <- 0
```

```
    if monster_appear['hp'] < 0 then
        oc_gain <- 5 + rng(25)
        output(f"Kamu berhasil mengalahkan
{monster_appear['type']}!")
        output(f"Kamu mendapatkan {oc_gain} OC!")
        win_battle <- true
    else if exit_battle then
        output("Kamu meninggalkan battle ini.")

    if not in_arena then
        -> oc_gain
    else if in_arena then
        -> win_battle, damage_dealt, damage_received
```

F09_Arena {Fitur untuk proses Arena ketika diinisiasi pada map}

```
function arena(bigdata: dict, user_id: int) -> none:
  total_damage_dealt <- 0
  total_damage_received <- 0

  repeat 5 times:
      output("WELCOME TO STAGE ", stage)
      time.sleep(3)
      win_battle, damage_dealt, damage_received <- battle(bigdata,
user_id <- user_id, in_arena <- true, level_setting <- stage)

      total_damage_dealt <- total_damage_dealt + damage_dealt
      total_damage_received <- total_damage_received +
damage_received

      if not win_battle then
            break
```

```
  output("damage dealt: ", total_damage_dealt)
  output("damage received: ", total_damage_received)
          else
            continue
```

F10_ShopCurrency {Fitur untuk mengelola OC pada Toko (Shop)}

```
procedure displayItems(bigdata, item_type) -> none:
    if item_type == "monster" then
        monster_temp <- bigdata['monster_shop'][1:]
        monster_shop <- []
        repeat monster in monster_temp:
            data <- monsterDetail(bigdata, monster_id <-
monster['monster_id'])
            // check = monster_temp[]
            id_temp <- data['id']
            repeat i in bigdata['monster_shop'] do
                if i['monster_id'] == id_temp then
                    temp_index <- bigdata['monster_shop'].index(i)
            data['stock'] <- monster_temp[temp_index-1]['stock']
            data['price'] <- monster_temp[temp_index-1]['price']
            monster_shop.append(data)

        output("ID | Type           | ATK Power | DEF Power | HP
| Stok | Harga")

output("==========================================================
========")
        repeat monster in monster_shop do
            output(f"{monster['id']:>2} | {monster['type']:<14} |
{monster['atk_power']:>9} | {monster['def_power']:>9} |
{monster['hp']:>4} | {monster['stock']:>4} |
{monster['price']:>6}")
```

```
    else if item_type == "potion" then
        item_temp <- bigdata['item_shop'][1:]
        item_shop <-[]
        repeat (length(item_temp)) times in id:
            temp <- {'id' : id+1}
            temp.update(item_temp[id])
            item_shop.append(temp)

        output("ID | Type                 | Stok | Harga")
        output("=========================================")
        repeat potion in item_shop do
            if potion['stock'] > 0 then
                output(f"{potion['id']:>2} | {potion['type']:<20}
| {potion['stock']:>4} | {potion['price']:>5}")

function buyItem(bigdata, user_id, item_type) -> dict:
    item_id <- convertToInt(input(f"Masukkan id {item_type}: "))
    oc <- bigdata['user'][user_id]['oc']
    if item_type == "monster" then
        // Cari monster yang sesuai dengan item_id
        monster <- next((m repeat m in bigdata['monster_shop'] if
m['monster_id'] == item_id), none)
        output(monster)
        // Jika monster ditemukan, cek apakah jumlah item yang
tersedia lebih dari 0
        // dan OC agen cukup
        if monster and monster['stock'] > 0 and oc >=
monster['price'] then
            // Jika benar, kurangi OC agen dan stok monster
            bigdata['monster_shop'][item_id]['stock'] <-
bigdata['monster_shop'][item_id]['stock'] - 1
            bigdata['user'][user_id]['oc'] <-
bigdata['user'][user_id]['oc'] - monster['price']

            monster_type <-
bigdata['monster'][monster['monster_id']]['type']
```

```
            // Tambahkan monster yang dibeli ke inventory agen
            output(f"Berhasil membeli monster: {monster_type}.
Monster sudah masuk ke inventory-mu!")
            temp <- {
                "user_id"    : user_id,
                "monster_id" : monster['monster_id'],
                'level'      : 1, }
            bigdata['monster_inventory'].append(temp)

            output(bigdata['monster_inventory'])
            output(bigdata['user'][user_id])
            -> bigdata
        else
            // Jika salah, berikan pesan error
            if not monster then
                output(f"Monster dengan ID {item_id} tidak
ditemukan.")
            else if monster['stock'] <= 0 then
                output(f"Monster
{bigdata['monster'][monster['monster_id']]['type']} dengan ID
{monster['monster_id']} tidak tersedia.")
            else
                output(f"OC-mu tidak cukup untuk membeli monster
{bigdata['monster'][monster['monster_id']]['type']}.")

            -> bigdata

    else if item_type == "potion" then
        quantity <- convertToInt(input(">>> Masukkan jumlah: "))

        // Cari potion yang sesuai dengan item_id
        potion <- bigdata['item_shop'][item_id]
        output(potion)
        // Jika potion ditemukan, cek apakah jumlah item yang
tersedia cukup
        // dan OC agen mencukupi
        if potion and potion['stock'] >= quantity and oc >=
```

```
potion['price'] * quantity then
            // Jika benar, kurangi OC agen dan stok potion
            bigdata['user'][user_id]['oc'] <-
bigdata['user'][user_id]['oc'] - potion['price'] * quantity
            // Reduce potion stock in bigdata
            potion['stock'] <- potion['stock'] - quantity
            // Tambahkan potion yang dibeli ke inventory agen
            add_potion <- next((p for p in
bigdata['item_inventory'] if (p['user_id'] == user_id and
p['type'] == potion['type'])), none)
            if add_potion then
                item_index <-
bigdata['item_inventory'].index(add_potion)
                bigdata['item_inventory'][item_index]['quantity']
<- bigdata['item_inventory'][item_index]['quantity'] + quantity
            else
                temp <- {
                        'user_id   : user_id,
                  'type'    : potion['type'],
                  'quantity'  : quantity
              }
              bigdata['item_inventory'].append(temp)

          output(f"Berhasil membeli item: {quantity}
{potion['type']}. Item sudah masuk ke inventory-mu!")
      else
          // Jika salah, berikan pesan error
          if not potion then
              output(f"Potion dengan ID {item_id} tidak
ditemukan.")
          else if potion['stock'] < quantity then
              output(f"Stok potion {potion['type']} dengan ID
{item_id} tidak mencukupi.")
          else
              output(f"OC-mu tidak cukup untuk membeli {quantity}
{potion['type']}.")
```

```
  // BuyItem(bigdata_example, 3, 'potion')
function shop(bigdata, user_id) -> dict:
  oc <- bigdata['user'][user_id]['oc']

  // combining monster data to one dictionary

  repeat:
      action <- input("Pilih aksi (lihat, beli, keluar)\n>>> ")
      if action not in ["lihat", "beli", "keluar"] then
          output("Input salah! Masukkan yang benar!")
      else if action.upper() == "LIHAT" then
          item_type <- input(">>> Mau lihat apa? (monster/potion):
")
          if item_type not in ["monster", "potion"] then
              output("Input salah! Masukkan yang benar!")
          else
              displayItems(bigdata, item_type)
      else if action.upper() == "BELI" then
          item_type <- input(">>> Mau beli apa? (monster/potion):
")
          output(f"Jumlah O.W.C.A. Coin-mu sekarang {oc}")
          if item_type not in ["monster", "potion"] then
              output("Input salah! Masukkan yang benar!")
          else
              buyItem(bigdata, user_id, item_type)
      else if action == "keluar" then
          output("Mr. Yanto bilang makasih, belanja lagi ya nanti
:)")
          exit repeat
  -> bigdata
```

F11 - Laboratory {Fitur untuk mewakili proses Laboratory}

```
function Laboratory(bigdata, user_id)
    Initialize
        cost <- [100, 300, 600, 1000]
        user_oc <- bigdata['user'][user_id]['oc']

    // Print all monsters
    user_monster <- UserInventory(user_id, bigdata,
'monster_inventory')
    Output("==EXAMPLE==")
    Output(user_monster, "\n")
    Output("============ MONSTER LIST ============")
    repeat each monster in user_monster
        Output(monster['monster_id'], "level: ", monster['level'])

    // Print upgrade costs
    Output()
    Output("1. Level 1 -> Level 2: 100 OC")
    Output("2. Level 2 -> Level 3: 300 OC")
    Output("3. Level 3 -> Level 4: 600 OC")
    Output("4. Level 4 -> Level 5: 1000 OC")
    Output()

    // Input monster
    choice <- int(input(">>> ")) - 1
    monster <- user_monster[choice]
    upgrade_cost <- cost[monster['level'] -1 ]
    if monster['level'] == 5 then
        Output("Maaf, monster yang Anda pilih sudah memiliki level
maksimum")
    else if
        Output(f"{monster['monster_id']} akan diupgrade menjadi
level {monster['level'] + 1}")
        Output(f"Harga untuk melakukan upgrade adalah
{upgrade_cost}")
        confirmation <- input(">>> Lanjutkan upgrade (Y/N): ")
        if confirmation == "Y" and user_oc >= upgrade_cost then
```

```
            monster['level'] += 1
            bigdata['user'][user_id]['oc'] <-
bigdata['user'][user_id]['oc'] - upgrade_cost
            Output("selamat")
        else
            Output("tidak berhasil")

        Output(bigdata['user'][user_id])
    else:
        Output(f"{monster['monster_id']} akan diupgrade menjadi
level {monster['level'] + 1}")
        Output(f"Harga untuk melakukan upgrade adalah
{upgrade_cost}")
        confirmation = input(">>> Lanjutkan upgrade (Y/N): ")
        if confirmation == "Y" and user_oc >= upgrade_cost then
            monster['level'] += 1
            bigdata['user'][user_id]['oc'] -= upgrade_cost
            Output("selamat")
        else:
            Output("tidak berhasil")

        Output(bigdata['user'][user_id])
    -> // return dalam notasi algoritma
```

F12 _ ShopManagement {Fitur untuk mengelola Toko (Shop)}

```
// ShopManagement
procedure ShopManagement(bigdata: dict)
    kond = True
    iterate
        action <- input(">>> Pilih aksi
(lihat/tambah/ubah/hapus/keluar): ")

        if action == 'lihat' then
```

```
            item_type <- input("monster/potion: ")
            DisplayItems(bigdata, item_type)
        else if action == 'tambah' then
            available_id <- []
            item_type <- input("monster/potion: ")

            if item_type == 'monster' then
                repeat length(bigdata['monster_shop']) - 1 times

available_id.append(bigdata['monster_shop'][i]['monster_id'])

                output("ID | Type           | ATK Power | DEF
Power | HP   ")

output("=====================================================")

                repeat for monster in bigdata['monster'][1:]
                    if monster['id'] not in available_id then
                        output(f"{monster['id']:>2} |
{monster['type']:<14} | {monster['atk_power']:>9} |
{monster['def_power']:>9} | {monster['hp']:>4}")

                monster_id <- int(input("Masukkan id monster: "))
                if monster_id in available_id then
                    output("ID telah ada di shop")
                else
                    monster_stock <- int(input("Masukkan stock
awal: "))
                    monster_price <- int(input("Masukkan harga:
"))
                    add_monster <- {
                        'monster_id': monster_id,
                        'stock': monster_stock,
                        'price': monster_price
                    }
                    bigdata['monster_shop'].append(add_monster)
                    output(bigdata['monster_shop'])
```

```
            else if item_type == 'potion' then
                // Display available potions and prompt user to
add a new potion
                // The implementation for potions will be similar
to monsters
                continue
        else if action == 'ubah' then
            item_type <- input("monster/potion:")

            if item_type == 'monster' then
                DisplayItems(bigdata, item_type)
                monster_id <- int(input("id: "))
                monster_stock <- int(input("stock: "))
                monster_price <- int(input("price: "))
                bigdata['monster_shop'][monster_id]['stock'] =
monster_stock
                bigdata['monster_shop'][monster_id]['price'] =
monster_price
            else if item_type == 'potion' then
                DisplayItems(bigdata, item_type)
                item_id <- int(input("id: "))
                item_stock <- int(input("stock: "))
                item_price <- int(input("price: "))
                bigdata['item_shop'][item_id]['stock'] =
item_stock
                bigdata['item_shop'][item_id]['price'] =
item_price
        else if action == 'hapus' then
            item_type <- input("monster/potion:")
            DisplayItems(bigdata, item_type)
            if item_type == 'monster' then
                monster_id <- int(input("id: "))
                delete_monster <- next((monster repeat monster in
bigdata['monster_shop'] if monster['monster_id'] == monster_id),
None)
                if delete_monster then
                    confirmation <- input(f"apakah anda yakin
```

```
ingin menghapus: ")
                    if confirmation == 'y' then
                        bigdata['monster_shop'] = [monster repeat
monster in bigdata['monster_shop'] if monster != delete_monster]
                else
                    output("Tidak ada id")
                continue
            else if item_type == 'potion' then
                item_id <- int(input("id: "))
                bigdata['item_shop'][item_id]['stock'] = 0
        else if action == 'keluar' then
            kond = False
        else
            output("Aksi tidak valid, silakan coba lagi")
    until kond == False
```

F13_MonsterManagement {Fitur untuk mengelola monster yang tersedia}

```
function MonsterManagement(bigdata: dict) -> dict
    repeat indefinitely:
        Output("SELAMAT DATANG DI MONSTER MANAGEMENT")
        Output("1. Tampilkan semua Monster")
        Output("2. Tambah Monster baru")
        Output("3. Keluar")
        choice <- int(input(">>> Pilih Aksi: "))
        if choice == 1 then
            Output("ID | Type           | ATK Power | DEF Power |
HP   ")

Output("=====================================================")
            repeat each monster in bigdata['monster'][1:]:
                Output(f"{monster['id']:>2} |
{monster['type']:<14} | {monster['atk_power']:>9} |
```

```
{monster['def_power']:>9} | {monster['hp']:>4}")
        else if choice == 2 then
            Output("Mulai pembuatan monster!!")
            available_type <- []
            repeat each monster in bigdata['monster'][1:]:
                available_type.append(monster['type'])
            repeat until (monster_type not in available_type):
                monster_type <- input("Masukkan nama monster: ")
                if monster_type in available_type then
                    Output("Nama sudah terdaftar, coba lagi!")
            monster_atk <- input(">>> Masukkan ATK Power: ")
            monster_atk <- int(monster_atk)
            monster_def <- input(">>> Masukkan DEF Power: ")
            monster_def <- int(monster_def)
            repeat until (monster_def >= 0 and monster_def <= 50):
                Output("DEF Power harus bernilai 0-50, coba
lagi!")
                monster_def <- input(">>> Masukkan DEF Power: ")
                monster_def <- int(monster_def)
            monster_hp <- input(">>> Masukkan HP: ")
            monster_hp <- int(monster_hp)
            Output("Monster baru berhasil dibuat")
            new_monster <- {
                'id'         : (bigdata['monster'][-1]['id'] + 1),
                'type'       : monster_type,
                'atk_power'  : monster_atk,
                'def_power'  : monster_def,
                'hp'         : monster_hp,
            }
            confirmation <- input(">>> Tambahkan Monster ke
database (Y/N): ")
            if confirmation == 'Y' then
                bigdata['monster'].append(new_monster)
                Output(bigdata['monster'])
            else if confirmation == 'N' then
                Output("Monster gagal ditambahkan")
            else
```

```
        else if choice == 3 then
            -> bigdata
        end
    end repeat
```

F14_Load {Fitur untuk proses load}

```
function CariFolder(nama_folder, address) -> boolean
    // Fungsi ini mencari folder pada direktori
    if nama_folder == '':
        output("Tidak ada nama folder yang diberikan!")
        -> False
    else:
        if os.path.exists(address):
            -> True
        else:
            output(f'\nFolder "{nama_folder}" tidak ditemukan.')
            -> False

function Load() -> tuple[boolean, dict]
    parser <- argparse.ArgumentParser() # membuat argument
    parser.add_argument('folder_simpan', help='folder tempat
tersimpan')
    args <- parser.parse_args() # wadah argumen
    address <- os.getcwd()
    xtrapath <- sys.argv[1]

    // validasi address
    valid <- False
```

```
    valid <- CariFolder(args.folder_simpan, address)
    if valid : // jika folder address ada
        output(text_ascii['loading1'])
        time.sleep(0.5)
        ClearScreen()
        output(text_ascii['loading2'])
        time.sleep(0.5)
        ClearScreen()
        output(text_ascii['loading3'])
        time.sleep(0.5)
        ClearScreen()
        output(text_ascii['loading4'])
        time.sleep(0.5)
        output()
        // membuat bigdata
        bigdata : dict <- {'user': [], 'monster': [],
'monster_shop': [], 'monster_inventory': [], 'item_shop': [],
'item_inventory': []}
        bigdata['user'] <- AmbilData(address + '/data/' + xtrapath
+ '/user.csv')
        bigdata['monster'] <- AmbilData(address + '/data/' +
xtrapath + '/monster.csv')
        bigdata['monster_shop'] <- AmbilData(address + '/data/' +
xtrapath + '/monster_shop.csv')
        bigdata['monster_inventory'] <- AmbilData(address +
'/data/' + xtrapath + '/monster_inventory.csv')
        bigdata['item_shop'] <- AmbilData(address + '/data/' +
xtrapath + '/item_shop.csv')
        bigdata['item_inventory'] <- AmbilData(address + '/data/'
+ xtrapath + '/item_inventory.csv')
        -> (True, bigdata)
    else: // jika masukan kosong
        output("Tidak ada nama folder yang diberikan!")
        -> (False, {})
```

F15_Save {FItur untuk proses Save}

```
function tulis_csv(data, folder, file) -> void
    f <- open(folder + '/' + file, "w")
    f.write(data)
    f.close

function save(bigdata: dict) -> void
    address <- input('Masukkan nama folder: ')
    output()

    iterate 4 times
        output('Saving' + '.' * i, end='\r')
        time.sleep(0.5)
    until i > 4

    if not os.path.isdir(address) then
        os.mkdir(address)

        iterate 4 times do
            output('Membuat folder data' + address + '.' * i,
end='\r')
            time.sleep(0.5)
        until i > 4
    end if

    // menyimpan data ke file
    tulis_csv((bigdata['user']), address, 'user.csv')
    tulis_csv((bigdata['monster']), address, 'monster.csv')
    tulis_csv((bigdata['monster_shop']), address,
'monster_shop.csv')
    tulis_csv((bigdata['monster_inventory']), address,
'monster_inventory.csv')
    tulis_csv((bigdata['item_shop']), address, 'item_shop.csv')
    tulis_csv((bigdata['item_inventory']), address,
'item_inventory.csv')
```

```
    output()
    output('Data telah disimpan pada folder ' + address + '!')
```

F16_Exit {Fitur untuk proses exit pada program}

```
function exit(bigdata: dict) -> void
    valid <- False
    iterate
        simpan <- input("Apakah Anda mau melakukan penyimpanan
file yang sudah diubah? (y/n) ")
        if simpan.lower() == 'y' then
            F15_Save.save(bigdata)
            valid <- True
        else if simpan.lower() == 'n' then
            valid <- True
        else
            valid <- False
    until valid == True
```

B03_Monsterball {Fitur untuk permainan monsterball}

```
function monsterBall(user_monster_pool, user_id, monster_id,
monster_level) -> list:
    seed <- RNG(100)
    temp <- {'user_id'    : user_id,
             'monster_id': monster_id,
             'level'      : monster_level,}

    if monster_level == 1 then
        if seed <= 75 then
            user_monster_pool.append(temp)
```

```
            output("Kamu berhasil menangkap")
            -> user_monster_pool
        else
            output("Kamu tidak berhasil menangkap")
            -> user_monster_pool

    else if monster_level == 2 then
        if seed <= 50 then
            user_monster_pool.append(temp)
            output("Kamu berhasil menangkap")
            -> user_monster_pool
        else
            output("Kamu tidak berhasil menangkap")
            -> user_monster_pool

    else if monster_level == 3 then
        if seed <= 25 then
            user_monster_pool.append(temp)
            output("Kamu berhasil menangkap")
            -> user_monster_pool
        else
            output("Kamu tidak berhasil menangkap")
            -> user_monster_pool

    else if monster_level == 4 then
        if seed <= 10 then
            user_monster_pool.append(temp)
            output("Kamu berhasil menangkap")
            -> user_monster_pool
        else
            output("Kamu tidak berhasil menangkap")
            -> user_monster_pool

    else if monster_level == 5 then
        if seed <= 5 then
            user_monster_pool.append(temp)
            output("Kamu berhasil menangkap")
```

```
            -> user_monster_pool
        else
            output("Kamu tidak berhasil menangkap")
            -> user_monster_pool
    else
        output("Level monster tidak valid")
        -> user_monster_pool
```

B04_Minigame {Fitur untuk memainkan Jackpot dan Hangman}

```
function playJackpot() -> None:
    output("")
    output("")
    output(" ASCII ART JACKPOT ”)
    output("")
    output("    ========== DAFTAR ITEM ===========")
    output("    1. Topi: 50 OC")
    output("    2. Pedang: 100 OC")
    output("    3. Koin: 200 OC")
    output("    4. Potion: 300 OC")
    output("    5. Monster: 500 OC ")
    output("")

    choice <- "Y"
    repeat
        choice <- input("Mulai bermain (Y/N): ").upper()
        if choice == "Y" then
            jackpot()
        else if choice == "N" then
            ->
    until choice == "Y"

function jackpot() -> int:
```

```
    jackpot_item <- {
        'topi'   : 50,
        'pedang' : 100,
        'koin'   : 200,
        'potion' : 300,
        'monster': 500,
    }

    ClearScreen()
    item_1 <- pickItem()

    output("---------------------------------------")
    output(f"---- {item_1} |         |         -----")
    output("---------------------------------------")

    sleep(RNG(2))
    ClearScreen()
    item_2 <- pickItem()
    output("---------------------------------------")
    output(f"---- {item_1} | {item_2} |        -----")
    output("---------------------------------------")

    sleep(RNG(2))
    ClearScreen()
    item_3 <- pickItem()

    output("---------------------------------------")
    output(f"---- {item_1} | {item_2} |  {item_3} -----")
    output("---------------------------------------")

    if item_1 == item_2 == item_3 then
        output("you get pokemon")
    else
        oc <- jackpot_item[item_1] + jackpot_item[item_2] +
jackpot_item[item_3]
        output(oc)
        -> oc
```

```
function pickItem() -> string:
    item <- RNG(5)
    if item == 1 then
        -> "topi"
    else if item == 2 then
        -> "pedang"
    else if item == 3 then
        -> "koin"
    else if item == 4 then
        -> "potion"
    else if item == 5 then
        -> "monster"
    else
```

B05_Peta_Kota_Danville {Fitur untuk mewakili peta kota danville!}

```
function create_agent(name, position) -> agent:
    -> {"name": name, "position": position}

function create_map(map_data) -> map:
    agent <- create_agent("Purry", (1, 1))  // Agen dimulai dari
posisi (1, 1)
    -> {"map_data": map_data, "agent": agent}

function print_map(map_data, agent_position) -> None:
    repeat  i <- 0 to length(map_data) - 1
        repeat  j <- 0 to length(map_data[i]) - 1
            if (i, j) = agent_position then
                output('P', end='')  // Cetak 'P' jika posisi sama
dengan posisi agen
            else
                output(map_data[i][j], end='')  // Cetak karakter
```

```
peta
        output()

function move_agent(map_data, agent_position, direction) -> (int,
int):
    x, y <- agent_position
    direction <- toLowerCase(direction)
    // Periksa batasan peta dan rintangan sebelum memindahkan agen
    if direction = "w" and x > 0 and map_data[x - 1][y] not in
['*', 'X', 'L', 'A', 'S'] then
        -> x - 1, y
    else if direction = "s" and x < length(map_data) - 1 and
map_data[x + 1][y] not in ['*', 'X', 'L', 'A', 'S'] then
        -> x + 1, y
    else if direction = "a" and y > 0 and map_data[x][y - 1] not
in ['*', 'X', 'L', 'A', 'S'] then
        -> x, y - 1
    else if direction = "d" and y < length(map_data[0]) - 1 and
map_data[x][y + 1] not in ['*', 'X', 'L', 'A', 'S'] then
        -> x, y + 1
    else
        -> agent_position

function check_location(map_data, agent_position) -> string:
    x, y <- agent_position
    places <- {'S': 'Shop', 'A': 'Arena', 'L': 'Laboratorium',
'X': 'Semak'}
    adjacent_places <- []
    // Periksa lokasi sekitar agen untuk area khusus
    if y + 1 < length(map_data[0]) and map_data[x][y + 1] in
places then
        append(adjacent_places, map_data[x][y + 1])
    if y - 1 >= 0 and map_data[x][y - 1] in places then
        append(adjacent_places, map_data[x][y - 1])
    if x + 1 < length(map_data) and map_data[x + 1][y] in places
then
        append(adjacent_places, map_data[x + 1][y])
```

```
    if x - 1 >= 0 and map_data[x - 1][y] in places then
        append(adjacent_places, map_data[x - 1][y])

    if length(adjacent_places) > 0 then
        output(places[adjacent_places[0]])
        output("Agen Purry akan mengakses ")
        output(join([places[place] repeat place in
adjacent_places]))
        output(", karena berada pada posisi yang bersebelahan
dengan ")
        output(join(adjacent_places))
        if places[adjacent_places[0]].toUpperCase() = 'SEMAK' then
            if RNG(5) = 5 then
                -> places[adjacent_places[0]]
            else
                -> "nothing"
        else
            -> places[adjacent_places[0]]
    else
        output("Agen Purry tidak berada di area khusus!")
        -> "nothing"

// peta_kota_danville
procedure peta_kota_danville(agent_position: tuple = (1, 1)) ->
tuple[str, tuple[int, int]]
    // Membaca data peta
    map_string <- """
************
*          *
*   S  X   *
*      X   *
*      X L *
* X  XXX   *
* X        *
* X        *
* XXX   A  *
*          *
```

```
*    XXXXX *
***********
"""
    map_data <- [list(line) repeat line in ManualSplit(map_string,
'\n') if line]

    iterate
        location <- "nothing"
        // Cetak peta
        ClearScreen()
        print_map(map_data, agent_position)
        output()

        // Cetak posisi agen
        output(f"Agen Purry di posisi: {agent_position}")

        // Periksa apakah agen berada di area khusus
        location <- check_location(map_data, agent_position)

        // Dapatkan input dari pengguna untuk navigasi
        direction <- input("Mau ke arah mana? (W/S/A/D/STOP)\n>>>
")

        if direction.upper() == "INVENTORY" then
            -> "Inventory", agent_position
        else if direction.upper() == "MONSTER" then
            -> "Monster", agent_position
        else if location.upper() == 'SEMAK' then
            -> location, agent_position
        else if location.upper() == direction.upper() then
            -> location, agent_position

        // Pindahkan agen
        if direction.upper() != "STOP" then
            agent_position <- move_agent(map_data, agent_position,
direction)
```

```
        // Periksa apakah pengguna ingin mengakhiri permainan
        else
            action <- input("Mau melakukan apa? (HELP/LOGOUT) ")
            if action.upper() == "LOGOUT" then
                -> "Logout", agent_position
            else if action.upper() == "HELP" then
                output("W: Ke atas (Utara)\nS: Ke bawah
(Selatan)\nA: Ke kiri (Barat)\nD: Ke kanan (Timur)\nEND: Keluar
dari permainan")
            else
                output("Melanjutkan perjalanan.")
    until False
```

DAFTAR PEMBAGIAN KERJA

Fitur	Implementasi	NIM Desainer	NIM Coder
F00_RNG	function RNG	16523109	16523109
F01_Register	function Registrasi function PilihMonster function PeriksaKPassword function PeriksaKUsername function PeriksaUsernameUnik procedure TambahAkunBaru	16523109	16523109
F02_Login	function login	16523109	16523109
F03_Logout	function logout	16523109	16523109
F04_Help	function help2	16523109	16523109
F05 - Monster	function NaikLevel function Attack function Defend	19623079	19623079
F06 - Potion	function PotionMenu function StrengthPotion function ResiliencePotion function HealingPotion	19623079	19623079
F07 - Monster	procedure DisplayMonsterStat procedure ShowInventory procedure DetailItem function InventoryUserDict function UserInventory	19623079	19623079
F08 - Battle	function PrintMonster function MonsterAppear function Battle	16523059	16523059
F09 - Arena	function Arena	16523059	16523059
F10 - Shop Currency	function DisplayItems function BuyItem function Shop	16523059	16523059
F11 - Laboratory	function Laboratory	19623019	19623019

F12 - Shop Management	procedure ShopManagement	19623019	19623019
F13 - Monster Management	function MonsterManagement	19623019	19623019
F14- Load	function load	19623259	19623259
F15 - Save	function tulis_csv function save	19623259	19623259
F16 - Exit	function exit	19623259	19623259
B03 - Monster Ball	function MonsterBall	19623079	19623079
B04 - Minigame	function Minigame function PlayJackpot function PlayHangman	19623079	19623079
B05 - Peta Kota Danville	function PrintMap function MoveAgent function CheckLocation procedure peta_kota_danville	16523109	16523109

Tabel 1 - Pembagian kerja

CHECKLIST HASIL RANCANGAN

Person In Charge	Fitur	Status Implementasi	Testing
William Anthony	F00 - RNG	Selesai	Selesai
William Anthony	F01 - Register	Selesai	Selesai
William Anthony	F02 - Login	Selesai	Selesai
William Anthony	F03 - Logout	Selesai	Selesai
William Anthony	F04 - Help	Selesai	Selesai
Barru Adi	F05 - Monster	Selesai	Selesai
Barru Adi	F06 - Potion	Selesai	Selesai
Barru Adi	F07 - Inventory	Selesai	Selesai
radhitia210205@gmail.com	F08 - Battle	Selesai	Selesai
radhitia210205@gmail.com	F09 - Arena	Selesai	Selesai
radhitia210205@gmail.com	F10 - Shop Currency	Selesai	Selesai
Michael Ballard I. S.	F11 - Laboratory	Selesai	Selesai
Michael Ballard I. S.	F12 - Shop Management	Selesai	Selesai
Michael Ballard I. S.	F13 - Monster Management	Selesai	Selesai
riyan rajab	F14 - Load	Selesai	Selesai
riyan rajab	F15 - Save	Selesai	Selesai
riyan rajab	F16 - Exit	Selesai	Selesai
Barru Adi	B03 - Monster Ball	Selesai	Selesai
Barru Adi	B04 - Minigame	Selesai	Selesai

Person In Charge	Fitur	Status Implementasi	Testing
William Anthony	**F00 - RNG**	Selesai	Selesai
William Anthony	**F01 - Register**	Selesai	Selesai
William Anthony	**F02 - Login**	Selesai	Selesai
William Anthony	**F03 - Logout**	Selesai	Selesai
William Anthony	**F04 - Help**	Selesai	Selesai
Barru Adi	**F05 - Monster**	Selesai	Selesai
Barru Adi	**F06 - Potion**	Selesai	Selesai
Barru Adi	**F07 - Inventory**	Selesai	Selesai
radhitia210205@gmail.com	**F08 - Battle**	Selesai	Selesai
radhitia210205@gmail.com	**F09 - Arena**	Selesai	Selesai
radhitia210205@gmail.com	**F10 - Shop Currency**	Selesai	Selesai
Michael Ballard I. S.	**F11 - Laboratory**	Selesai	Selesai
Michael Ballard I. S.	**F12 - Shop Management**	Selesai	Selesai
Michael Ballard I. S.	**F13 - Monster Management**	Selesai	Selesai
William Anthony	**B05 - Map**	Selesai	Selesai
Barru Adi	**main.py**	Selesai	Selesai

*Catatan: Semua anggota melakukan testing, sehingga NIM Tester terdiri dari semua anggota.

LAMPIRAN

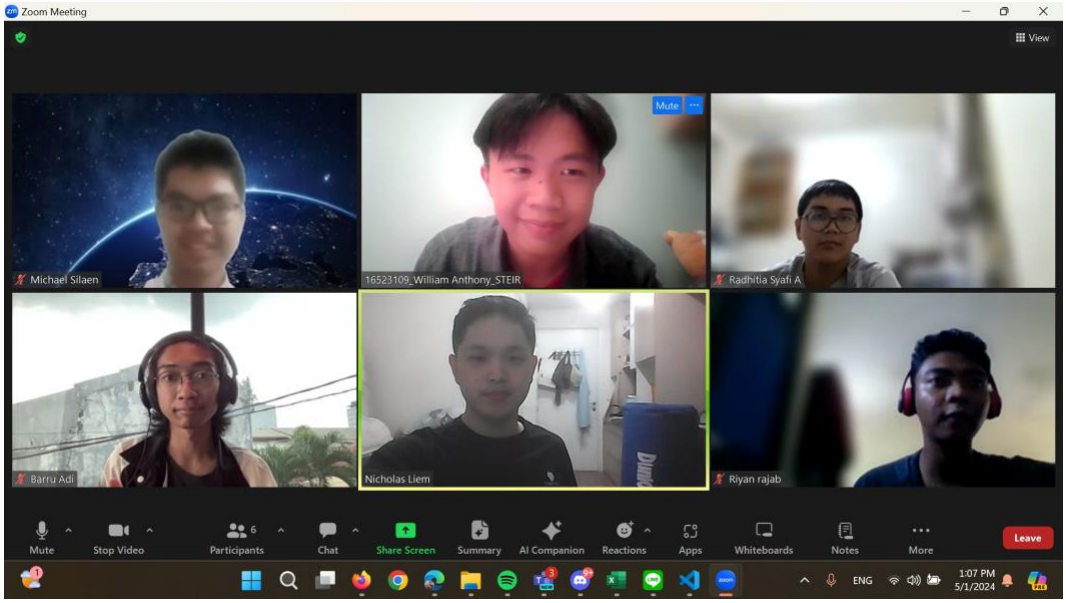
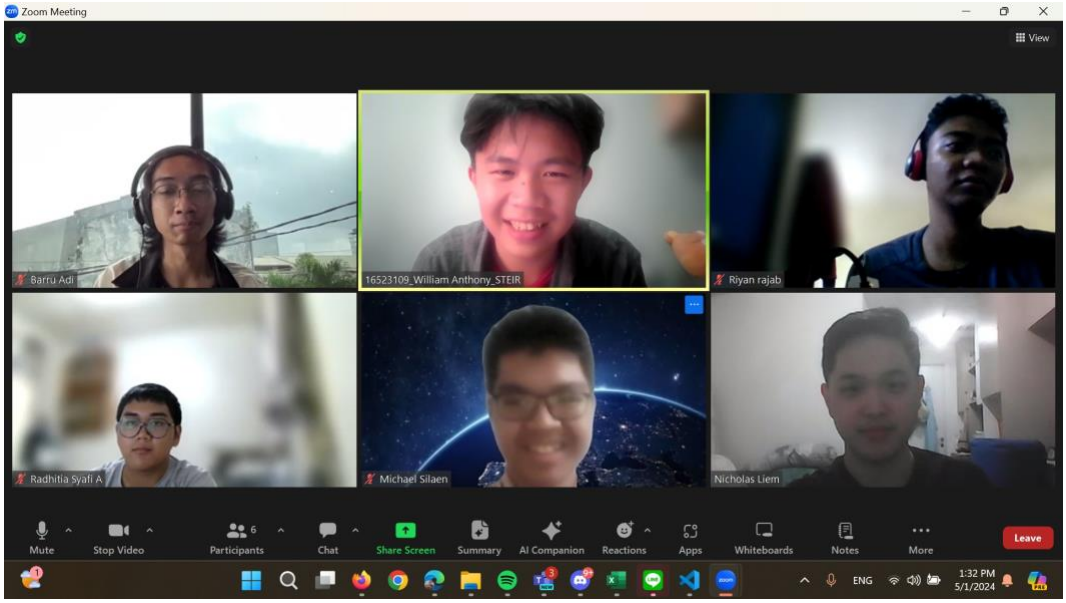
Asistensi-1

Nomor Asistensi	**:**	**1**
No. Kelompok/Kelas	**:**	**E/K-10**
Tanggal asistensi	**:**	**Rabu, 1 Mei 2024**

Anggota kelompok		**NIM / Nama (Hanya yang Hadir)**
	1	**16523109 / William Anthony**
	2	**16523059 / Radhitia Syafi Alfardzan (Notulen)**
	3	**19623259 / M. Riyan Rajab Setiawan**
	4	**19623079 / Barru Adi Utomo**
	5	**19623019 / Michael Ballard Isaiah Silaen**
	6	**-**
Asisten pembimbing		**NIM / Nama**
		13521135 / Nicholas Liem

Catatan Asistensi:

Rangkuman Diskusi
Headline : Definisi dan Spesifikasi Ataupun Batasan Yang Diperbolehkan - Radhitia Syafi Alfardzan **- Kak Nicholas Liem :** F00 Satu File satu fungsi, misal RNG pakai LCG terserah implementasinya dan gaboleh pake library nya. **- William Anthony :** Apakah boleh pemakaian chatgpt agar mempermudahkan untuk lebih memahami masalahnya? **Kak Nicholas Liem :** Menurut aku sendiri sih tidak boleh, lebih baik liat github orang bagaimana dan bisa liat github kakak dan nanti kalian cari di paling ujung page dan cara mengimport csv pakai import argparse. Berikut link kakak: **https://github.com/NicholasLiem/IF1210_TugasBesar_BNMO-Toko-Game/blob/main/load.py** **- William Anthony :** Masih bisa ga pakai fungsi? **Kak Nicholas Liem :** Iya tentunya boleh, malah dari fungsi sebelumnya yang dari LCG **- Barru Adi :** Kita kan dibatasi untuk split ya mau buat sendiri boleh kak? **Kak Nicholas Liem :** Boleh2, dulu aku buat sendiri juga **- Kak Nicholas Liem :** Untuk penjelasan GIT ada link berikut Record Tubes Daspro

Tindak Lanjut
Tindak Lanjut Spesifikasi Tugas Besar dan Pembagian Tugas Kode dan Laporan Saling Melengkapi
Dokumentasi
Awal :    **Akhir :**    **MoM : 25 Menit**

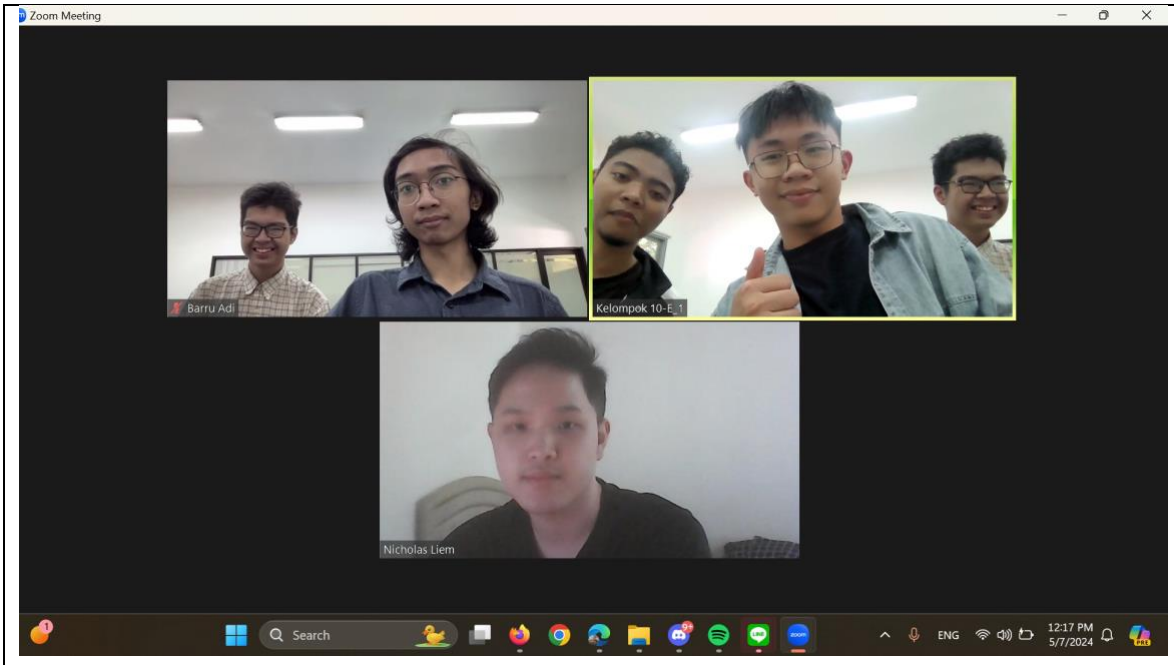
Asistensi-2

Nomor Asistensi	**:**	**2**
No. Kelompok/Kelas	**:**	**E/K-10**
Tanggal asistensi	**:**	**Selasa, 7 Mei 2024**

Anggota kelompok		**NIM / Nama (Hanya yang Hadir)**
	1	**16523109 / William Anthony**
	2	**16523059 / Radhitia Syafi Alfardzan**
	3	**19623259 / M. Riyan Rajab Setiawan**
	4	**19623079 / Barru Adi Utomo**
	5	**19623019 / Michael Ballard Isaiah Silaen**
	6	**-**
Asisten pembimbing		**NIM / Nama**
		13521135 / Nicholas Liem

Catatan Asistensi:

Rangkuman Diskusi
Headline : Definisi dan Spesifikasi Laporan - William Anthony **- Kak Nicholas Liem : Tidak ada ketentuan yang ketat namun, berikut merupakan referensi.** **- Barru Adi Utomo : Apakah boleh menggunakan variabel global untuk pengerjaan daspro?** **Kak Nicholas Liem : Tidak dianjurkan tapi boleh. Sangat dianjurkan untuk coba sendiri dahulu**
Tindak Lanjut
Tindak Lanjut Spesifikasi Tugas Besar dan Pembagian Tugas Kode dan Laporan Saling Melengkapi
Dokumentasi
Awal :



Akhir :

MoM : 10 Menit