LAPORAN TUGAS BESAR

MATA KULIAH IF1201 - DASAR PEMROGRAMAN

PROGRAM “RENCANA KESELAMATAN KOTA DANVILLE”

KELAS 08

KELOMPOK K08-A

DISUSUN OLEH:

Syifannissa Nafisalia Kuncoro	16523048
Julian Benedict	16523178
Jason Samuel	19623118
Nadhif Radityo Nugroho	19623178
Aryo Wisanggeni	19623258

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

HALAMAN PERNYATAAN

"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."

Syifannissa Nafisalia Kuncoro	16523048
Julian Benedict	16523178
Jason Samuel	19623118
Nadhif Radityo Nugroho	19623178
Aryo Wisanggeni	19623258

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

I. DESKRIPSI PERSOALAN

Sebagai agen-agen di O.W.C.A (Organisasi Warga Cool Abiez), atas permintaan Purry, kita harus bekerja sama untuk mengalahkan monster-monster terbaru Dr. Asep Spakbor. Untuk mencapai tujuan itu, kita harus merancang rencana yang matang dan mencari serta melatih monster-monster sendiri untuk melawan monster-monster Dr. Asep Spakbor demi keselamatan kota Danville.

II. PEMBAGIAN TUGAS

Tabel II.I Pembagian Tugas Berdasarkan Fitur

Fitur	Implementasi	NIM Designer	NIM Coder	NIM Tester
F00 – Random Number Generator	Fungsi rand()	19623258	19623258	19623258
F01 – Register	Prosedur user_register()	19623258	19623258	19623178
F02 – Login	Prosedur user_login()	19623258	19623258	19623178
F03 – Logout	Prosedur user_logout()	19623258	19623258	19623178
F04 – Menu & Help	Prosedur Help()	16523178	16523178	16523178
F05 – Monster	Database monster.csv dan monster_inventory.csv	16523048	16523048	16523048
F06 – Potion	Tipe PotionSchemaType	19623118	16523178 19623178	16523178 19623178
F07 – Inventory	Program inventory.py	19623118	19623118	19623178
F08 – Battle	Fungsi battle_get()	19623178	19623178	19623178
F09 – Arena	Fungsi arena()	19623178	19623178	19623178
F10 – Shop & Currency	Fungsi shop() dan database user.csv	16523178	16523178	16523178
F11 – Laboratory	Fungsi laboratory()	16523178	16523178	16523178
F12 – Shop Management	Fungsi shop() diakses sebagai admin	16523048	16523048	16523048
F13 – Monster Management	Fungsi Monster()	16523048	16523048	16523048
F14 – Load	Prosedur database_load()	19623178	19623178	19623178
F15 – Save	Prosedur database_save()	19623178	19623178	19623178
F16 – Exit	Prosedur exit()	19623118	19623118	19623118

Tabel II.II Checklist Hasil Rancangan

Fitur	Desain	Implementasi	Testing
F00 – Random Number Generator	✓	✓	✓
F01 – Register	✓	✓	✓
F02 – Login	✓	✓	✓
F03 – Logout	✓	✓	✓
F04 – Menu & Help	✓	✓	✓
F05 – Monster	✓	✓	✓
F06 – Potion	✓	✓	✓
F07 – Inventory	✓	✓	✓
F08 – Battle	✓	✓	✓
F09 – Arena	✓	✓	✓
F10 – Shop & Currency	✓	✓	✓
F11 – Laboratory	✓	✓	✓
F12 – Shop Management	✓	✓	✓
F13 – Monster Management	✓	✓	✓
F14 – Load	✓	✓	✓
F15 – Save	✓	✓	✓
F16 – Exit	✓	✓	✓

Keterangan: ✓ : sudah selesai dikerjakan

X : sudah dikerjakan, tetapi tidak selesai

- : tidak dikerjakan sama sekali

III. DESAIN COMMAND

Tabel III.I – Desain Command Berdasarkan Fitur

Fitur	Nama Command	Input	Output
F01 – Register	register	• new_username: string • new_password: string • starter_choice: string (‘1’, ‘2’, atau ‘3’)	User membuat akun baru dan langsung ke-login
F02 – Login	login	• username: string • password: string	User dalam kondisi login
F03 – Logout	logout	None	User dalam kondisi logout
F04 – Menu & Help	help	None	Tampilan Menu & help (String) sesuai dengan agent atau admin atau belum login
F07 – Inventory	inventory	• input1 : integer	Tampilan inventory (Integer & string)
F08 – Battle	battle	• selfMonster: integer • user_action: string	Tampilan menang jika menang dan oc bertambah, tampilan kalan jika kalah
F09 – Arena	arena	• selfMonster: integer • user_action: string	Tampilan menang jika menang dan oc bertambah, tampilan kalah jika kalah
F10 – Shop & Currency	shop	• Aksi : stringAksi_2: string • Aksi_3: string • Aksi_beli_monster: string • Aksi_beli_potion: integer • Jumlah_beli_potion: integer • Beli_ga : String	Tampilan monster yang dibeli dan potion yang dibeli (string)

F11 – Laboratory	laboratory	• Choice : integer • Lanjut_upgrade: string	Tampilan monster dan level (string)
F12 – Shop Management	shop	• aksi : integer • id : integer • type : string • monsters : array of integer and integer • item_shop : array of string and integer • stokAwal : integer • harga : integer • stokBaru : integer • hargaBaru : integer • yakin : string	Tampilan monsters dan potion (integer & string)
F13 – Monster Management	monster	• aksi : integer • def : integer • type : string • ATKpower : integer • HP : integer • masukkan : string • monsters : array of integer & string	Tampilan monsters (integer & string)
F14 – Load	load	• nama_folder: string	Data dari folder telah di-load
F15 – Save	save	• nama_folder: string	Data telah ke-save ke folder
F16 – Exit	exit	• input1 : string • A : string	User exit dari program

IV. DESAIN KAMUS DATA

User.csv

type rekamanUser : <id: integer, username: string, password: string, role: string (“agent” or “admin”), oc: integer>

constant markUser : rekamanUser = “”

user : SEQFILE of

(*) RekUser : rekamanUser

(1) markUser

Monster.csv

type rekamanMonster : <id: integer, type: string, atk_power: integer, def_power: integer, hp: integer>

constant markMonster : rekamanMonster = “”

monster : SEQFILE of

(*) RekMonster : rekamanMonster

(1) markMonster

Monster_shop.csv

type rekamanMonsterShop : <monster_id: integer, stock: integer, price: integer>

constant markMonsterShop : rekamanMonsterShop= “”

monsterShop : SEQFILE of

(*) RekMonsterShop : rekamanMonsterShop

(1) markMonsterShop

Monster_inventory.csv

type rekamanMonsterInventory : <user_id: integer, monster_id: integer, level: integer>

constant markMonsterInventory : rekamanMonsterInventory= “”

monsterInventory : SEQFILE of

(*) RekMonsterInventory : rekamanMonsterInventory

(1) markMonsterInventory

Item_shop.csv

type rekamanItemShop : <type: string, stock: integer, price: integer>

constant markItemShop : rekamanItemShop= “”

itemShop : SEQFILE of

(*) RekItemShop: rekamanItemShop

(1) markItemShop

Item_inventory.csv

type rekamanItemInventory : <user_id: integer, monster_id: integer, level: integer>

constant markItemInventory : rekamanItemInventory= “”

itemInventory : SEQFILE of

(*) RekItemInventory : rekamanItemInventory

(1) markItemInventory

V. DESAIN DEKOMPOSISI ALGORITMIK DAN FUNGSIONAL PROGRAM

Dekomposisi F01 - REGISTER

{I.S. : data user dari database dan data login tersedia}

Mengecek apakah user dalam keadaan login atau tidak

{I.S./F.S. : mengecek apakah sudah ada user yang sedang login atau tidak, jika iya, register berhenti di sini}

Input username baru

{I.S./F.S. : user menginput username untuk akun barunya}

Mengecek apakah karakter username baru valid

{I.S./F.S. : mengecek apakah username berisi karakter-karakter yang valid, yaitu alfabet, angka, underscore, dan strip, jika tidak, register berhenti di sini}

Mengecek apakah username baru sudah ada di didatabase

{I.S./F.S. : mengecek apakah username sudah dipakai oleh user lain di database, jika iya, register berhenti di sini}

Memilih monster pertama

{I.S./F.S. : user menginput 1, 2, atau 3 untuk memilih monster pertamanya, jika tidak menginput dengan benar, akan diprompt berkali-kali hingga user menginput yang valid}

{F.S. : data user baru direkam ke user.csv dan monster pertama disimpan ke user baru di inventory.csv}

Dekomposisi F02 - LOGIN

{I.S. : data user dari database dan data login tersedia}

Mengecek apakah user dalam keadaan login atau tidak

{I.S./F.S. : mengecek apakah sudah ada user yang sedang login atau tidak, jika iya, login berhenti di sini}

Input username

{I.S./F.S. : user menginput username akunnya}

Mengecek apakah user sudah ada di database

{I.S./F.S. : mengecek apakah username yang diinput sudah ada di database, jika tidak, login berhenti di sini}

Input password

{I.S./F.S. : user menginput password akunnya}

Mengecek apakah password benar

{I.S./F.S. : mengecek apakah password yang diinput sudah sesuai dengan akunnya atau tidak, jika tidak, login berhenti di sini}

{F.S. : user dalam keadaan login sesuai dengan username yang diinput}

Dekomposisi F03 - LOGOUT

{I.S. : data login tersedia}

Mengecek apakah user dalam keadaan login atau tidak

{I.S./F.S. : mengecek apakah sudah ada user yang sedang login atau tidak, jika tidak, logout berhenti di sini}

{F.S. : user keluar dari keadaan login}

Dekomposisi F04 – MENU & HELP

{I.S. : data login tersedia }

Menampilkan help berdasarkan keadaan login user

{I.S./F.S. :

1. jika user tidak login, menampilkan help untuk user yang belum login

2. jika user sudah login, menampilkan help untuk agent jika user login sebagai agent dan menampilkan help untuk admin jika user login sebagai admin}

{F.S. : menu help berdasarkan keadaan login user tertampil }

Dekomposisi F07 – INVENTORY

{I.S. : Data user, monster inventory, dan item inventory tersedia}

Mengambil data dari csv

{I.S./F.S. : mengambil data user, monster inventory, dan item inventory dari database}

Mengecek apakah id player yang aktif sesuai dengan database

{I.S./F.S. : jika id player yang aktif sesuai dengan database save maka akan ditampilkan inventory milik player tersebut, jika tidak inventory berhenti di sini }

Menampilkan inventory milik player

{I.S./F.S. : ditampilkan inventory milik player tersebut }

{F.S. : jika diinput angka, akan menampilkan detail dari item di inventory}

Dekomposisi F08 – BATTLE

{I.S. : data monster tersedia}

Generasi monster random

{I.S./F.S. : mengambil monster random dari database monster, serta dirandomkan levelnya juga}

User input monster monster yang ingin digunakan dalam battle

{I.S./F.S. : user menginput monster yang ingin digunakan dalam battle, jika monster yang dipilih tidak ada, user diminta untuk input lagi hingga input valid }

User pilih perintah untuk aksi monster

{I.S./F.S. : user menginput perintah yang ingin dilakukan monster dalam battle }

{F.S. : user menang ketika mengalahkan monster lawan, user kalah ketika semua monsternya habis }

Dekomposisi F09 – ARENA

{I.S. : data inventory monster tersedia}

User input monster monster yang ingin digunakan dalam battle

{I.S./F.S. : user menginput monster yang ingin digunakan dalam battle, jika monster yang dipilih tidak ada, user diminta untuk input lagi hingga input valid }

Generasi monster random

{I.S./F.S. : tipe dan level monster lawan random. Generasi monster ini dilakukan maksimal lima kali per satu ronde arena }

{F.S. : user menang ketika mengalahkan semua lima monster lawan dan menerima oc, user kalah ketika dikalahkan salah satu dari lima monster lawan }

Dekomposisi F10 – SHOP & CURRENCY

{I.S. : data user yang login dan data shop tersedia}

User pilih aksi

{I.S./F.S. : user memilih aksi untuk melihat, beli, atau keluar dari shop, jika keluar, shop berhenti }

Lanjutan jika user pilih lihat

{I.S./F.S. : user memilih untuk melihat item atau monster, pilih item maka program akan menampilkan item yang dijual serta spesifikasinya, pilih monster maka program akan menampilkan monster yang dijual serta spesifikasinya}

Lanjutan jika user pilih beli

{I.S./F.S. : ditampilkan jumlah oc yang dimiliki user, serta diberi pilihan apakah user ingin membeli item atau monster, selanjutnya user perlu input id monster/item yang ingin dibeli }

Mengecek apakah user mempunyai oc yang cukup

{I.S./F.S. : jika oc yang dimiliki cukup untuk membeli, pembelian sukses, jika tidak, pembelian gagal dan user kembali untuk menginput lihat, beli, atau keluar dari shop }

{F.S. : data inventory di update dengan hasil pembelian user}

Dekomposisi F11 – LABORATORY

{I.S. : data monster inventory dan user tersedia}

Ditampilkan informasi monster dan user

{I.S./F.S. : ditampilkan informasi monster yang dimiliki user serta level dan spesifikasinya, serta ditampilkan oc yang dimiliki user }

User input monster yang ingin diupgrade

{I.S./F.S. : user menginput id monster yang ingin diupgrade sesuai id yang ditampilkan }

Konfirmasi menaikan level monster

{I.S./F.S. : user menginput (y/n) untuk mengonfirmasi apakah monster akan dinaikan levelnya atau tidak}

Mengecek apakah level monster sudah maks atau tidak

{I.S./F.S. : jika tidak, laboratory lanjut, jika iya, laboratory berhenti di sini}

Mengecek apakah user punya oc yang cukup

{I.S./F.S. : jika oc cukup, upgrade sukses, jika tidak, upgrade tidak sukses}

{F.S. : data monster inventory ter-update dengan perubahan yang dilakukan user }

Dekomposisi F12 – SHOP MANAGEMENT

{I.S. : data shop tersedia}

Mengecek apakah user dalam keadaan login atau tidak

{I.S./F.S. : mengecek apakah sudah ada user yang sedang login atau tidak, jika iya, login berhenti di sini}

Input aksi yang ingin dilakukan

{I.S. : User menginput aksi, jika aksi tidak sesuai maka program akan berhenti}

Input id, stok awal, harga atau id, stok baru, harga baru

{I.S.. : User menginput aksi, jika aksi tidak sesua maka program akan berhenti}

{F.S. : Program akan memunculkan atau menambah atau mengubah atau menghapus data dari potion atau monster.}

{F.S. : data shop terbarukan sesuai update yang dilakukan user }

Dekomposisi F13 – MONSTER MANAGEMENT

{I.S. : data monster tersedia}

Mengecek apakah user login sebagai admin atau tidak

{I.S./F.S. : mengecek apakah user login sebagai admin atau tidak, jika tidak, program berhenti di sini}

Input aksi yang ingin dilakukan

{I.S.: User menginput aksi, jika input tidak sesuai, program akan berhenti disini}

{F.S. : Jika input = 1, menampilkan semua monster. Jika input = 2, menambah monster baru, lalu menampikan data monster yang baru}

{F.S. : data monster diupdate sesuai dengan aksi yang dilakukan}

Dekomposisi F14 – LOAD

{I.S. : folder data save game tersedia}

Pengguna memulai program dengan mencantum nama folder save

{I.S./F.S. : mengecek apakah folder save yang dicantumkan ada atau tidak, jika tidak, load berhenti di sini}

{F.S. : game dengan keadaan persis dengan folder save telah di-load}

Dekomposisi F15 – SAVE

{I.S. : folder data save game tersedia}

Pengguna input nama folder untuk tujuan save

{I.S./F.S. : pengguna menginput nama folder untuk tujuan savenya}

Mengecek apakah folder tujuan ada atau tidak

{I.S./F.S. : mengecek apakah folder tujuan ada atau tidak, jika iya, data akan di-save ke folder yang sudah ada, jika tidak, folder baru akan dibuat dengan nama yang diinput, serta data akan di-save ke folder baru tersebut}

{F.S. : data game ke=save ke folder tujuan}

Dekomposisi F16 – EXIT

{I.S. : data folder save tersedia}

Mengecek apakah user ingin save sebelum exit

{I.S./F.S. : mengecek apakah user ingin save sebelum exit, jika iya, maka akan melakukan save lalu system exit. Jika tidak, maka akan langsung system exit. Jika input tidak valid, maka akan menanyakan ulang.}

{F.S. : keluar dari program }

VI. SPESIFIKASI MODUL, PROSEDUR, DAN FUNGS

Program Berdasarkan Fitur

```
Program user
{ Program yang berisi fungsi-fungsi yang berkaitan dengan authentication user }
KAMUS
Type GameState: typed dictionary
                    <"GameState",
                       userDatabase: Database[UserSchemaType]
                       monsterDatabase: Database[MonsterSchemaType]
                        potionDatabase: Database[PotionSchemaType]
                       battleDatabase: Database[BattleSchemaType]
                       arenaDatabase: Database[ArenaSchemaType]
                       shopDatabase: Database[ShopSchemaType]
                       laboratoryDatabase: Database[LaboratorySchemaType]
                       inventoryItemDatabase: Database[InventoryItemSchemaType]
                       inventoryMonsterDatabase: Database[InventoryMonsterSchemaType]
                       userId: integer or None
                       visual: typed dictionary<"Visual",
                                                driver: Driver,
                                                toplevel: TopLevel,
                                                view: View or None,
                                                directory: str,
                                                splashes: dictionary[string, list[list[Rune]]]>>

type UserSchemaType: named tuple<"User",
                                          [("id": integer),
                                          ("username": string),
                                          ("password": string),
                                          ("role": string ("admin", "agent", atau "system"),
                                          ("money", float)]>

function user_is_logged_in (gameState : GameState) → Boolean
{ menghasilkan boolean (true/false) berdasarkan apakah terdapat user yang login atau
tidak, true jika iya, false jika tidak}

function user_get_current(gameState: GameState) → UserSchemaType
{ mengambil data pengguna yang login }

procedure user_register(gameState: GameState) → None
{ registrasi user baru }

procedure user_login(gameState: GameState, username: str, password: str) → None:
{ login user ke akunnya }

procedure user_logout(gameState: GameState) -> None:
{ logout user dari keadaan login }
```

ALGORITMA PROGRAM UTAMA

```
function user_is_logged_in (gameState : GameState) → Boolean
{ menghasilkan boolean (true/false) berdasarkan apakah terdapat user yang login atau
tidak, true jika iya, false jika tidak}
KAMUS LOKAL
ALGORITMA
   Output gamestate_get_user_id(gameState) is not None

function user_get_current(gameState: GameState) → UserSchemaType
{ mengambil data pengguna yang login }
KAMUS LOKAL
   userId: integer
   user_database: UserSchemaType
ALGORITMA
   userId ← gamestate_get_user_id(gameState)
   If userId is None then
      output None
   user_database ← gamestate_get_user_database(gameState)
   output database_get_entry_at(user_database, userId)

procedure user_register(gameState: GameState) → None
{ registrasi user baru }
KAMUS LOKAL
  user_database: UserSchemaType
  user_entries: list of tuple
  user_false_input: boolean
  valid_username: string
  new_password: string
  monster_false: boolean
  input_salah: boolean
  starter_choice: string
  new_id: integer
ALGORITMA
   user_database ← gamestate_get_user_database(gameState)
   user_entries ← database_get_entries(user_database)

   { mengecek apakah sudah ada user yang login atau tidak }
   if user_is_logged_in(gameState) then
     output "Anda sudah login, logout dulu untuk register!"

   { input username baru}
   input new_username
   user_false_input ← True
   valid_username ←
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ-_1234567890"

   { mengecek apakah karakter dalam username baru valid }
   i traversal [0 .. length(new_username)-1]
     j traversal [0 .. length(valid_username)-1]
```

```
        if new_username[i] ≠ valid_username[j] then
            user_false_input ← True
        else
            user_false_input ← False
            break
        if user_false_input ← True then
          break

   if user_false_input = True then
      output "Username hanya boleh berisi alfabet, angka, underscore, dan strip!"
   else
      { mengecek apakah sudah ada username tersebut dalam database }
      i traversal [0 .. len(user_entries)-1]
         if new_username = user_entries[i].username then
            output "Username {new_username} sudah terpakai, silahkan gunakan username
lain!")

   { input password baru }
   input new_password

   { menginput pilihan monster pertama user baru }
   monster_false ← True
   input_salah ← False
   while monster_false = True do
      if input_salah = True then
          input starter_num
          if starter_num = "1" or starter_num = "2" or starter_num = "3" then
             monster_false ← False
          else
             input_salah ← True

   if starter_num = '1' then
      starter_choice ← "Monster1"
   else if starter_num = '2' then
      starter_choice ← "Monster2"
   else
      starter_choice ← "Monster3"

  { write data user baru ke database dan langsung login user baru }
   new_id ← database_get_entries_length(user_database)
   database_set_entry_at(user_database, new_id,
                                UserSchemaType(
                                id = new_id,
                                username = new_username,
                                password = new_password,
                                role = "agent",
                                money = 0))
   gamestate_set_user_id(gameState, new_id)
   output
```

```
procedure user_login(gameState: GameState, username: str, password: str) → None
{ login user ke akunnya }
KAMUS LOKAL
    user_database: UserSchemaType
    user_ entries: list of tuple
    username: string
    user_DNE: boolean
    user_index: integer
    password: string
ALGORITMA
   user_database ←gamestate_get_user_database(gameState)
   user_entries ← database_get_entries(user_database)

   { mengecek apakah sudah ada user yang login atau tidak }
   if user_is_logged_in(gameState) then
      output "Anda sudah login, logout dulu untuk login!")

   { input username }
   input username

   { mencari apakah user ada }
   user_DNE ← True
   i traversal [0 .. length(user_entries)-1]
      if username = user_entries[i].username then
         user_DNE ← False
         user_index ← i
         break

   { jika user yang diinput tidak ada, akan exit dari prosedur }
   if user_DNE = True then
      output "Username tidak terdaftar!"

   { input password }
   input password

   { mengecek password benar atau tidak }
   if password ≠ user_entries[user_index].password then { password salah }
      output “Password salah!"
   else: { password benar }
     newly_logged_in_id = user_entries[user_index].id
     gamestate_set_user_id(gameState, newly_logged_in_id)

      output '"
Selamat datang, Agent {username}!
Masukkan command “help” untuk daftar command yang dapat kamu panggil.
          '"

procedure user_logout(gameState: GameState) → None
{ logout user dari keadaan login }
KAMUS LOKAL
```

```
ALGORITMA
    { mengecek apakah user sedang login atau tidak }
    if gamestate_get_user_id(gameState) is None then
        output “Anda belum login, login dulu untuk logout!"
    else
        gamestate_set_user_id(gameState, None)
        output “Anda sukses logout”
```

```
Program Menu and help
{ berisi fingsi/prosedur untuk fitur F04}
{ Algoritma ini menampilkan pesan bantuan berdasarkan peran pengguna }
```

```
KAMUS
  user_role: string { Peran pengguna, dapat berupa 'Agent', 'Admin', atau None }
  user_name: string { Nama pengguna }
  user_database: data_structure { Basis data pengguna }
  user_entry: data_structure { Entri pengguna dari basis data }

prosedur:
  help(user_role: string) { Menampilkan pesan bantuan berdasarkan peran pengguna }
```

```
ALGORITMA PROGRAM UTAMA
{ Algoritma ini menampilkan pesan bantuan berdasarkan peran pengguna }
    if user_role = None 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")
    output("")
    output("Footnote:")
    output("1. Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang
terdaftar")
    output("2. Jangan lupa untuk memasukkan input yang valid")
  else if user_role = "Agent" then
    output("========== HELP ==========")
    output("Halo Agent ", user_name, ". 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. Inventory: Melihat owca-dex yang dimiliki oleh Agent")
    output("3. Battle: Untuk melawan monster musuh secara random dan jika menang akan
mendapatkan OC (Uang)")
    output("4. Laboratory: Untuk meningkatkan level dari monster yang kalian punya")
    output("5. Shop: Untuk membeli Monster dan juga potion!!")
    output("")
    output("Footnote:")
```

```
    output("1. Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang
terdaftar")
    output("2. Jangan lupa untuk memasukkan input yang valid")
  else if user_role = "Admin" then
    output("========== HELP ==========")
    output("Selamat datang, Admin. Berikut adalah hal-hal yang dapat kamu lakukan:")
    output("1. Logout: Keluar dari akun yang sedang digunakan")
    output("2. Shop: Melakukan manajemen pada SHOP sebagai tempat jual beli peralatan
Agent")
    output("3. Monster: Melakukan manajemen pada Monster, seperti menambahkan
monster ke shop, dan cek database monsternya")
    output("")
    output("Footnote:")
    output("1. Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang
terdaftar")
    output("2. Jangan lupa untuk memasukkan input yang valid")

AKHIR ALGORITMA help
```

```
{ REALISASI FUNGSI/PROSEDUR }
```

```
Program inventory
{ berisi fungsi/prosedur inventory }
KAMUS

Type GameState: typed dictionary
                    <"GameState",
                       userDatabase: Database[UserSchemaType]
                       monsterDatabase: Database[MonsterSchemaType]
                        potionDatabase: Database[PotionSchemaType]
                       battleDatabase: Database[BattleSchemaType]
                       arenaDatabase: Database[ArenaSchemaType]
                       shopDatabase: Database[ShopSchemaType]
                       laboratoryDatabase: Database[LaboratorySchemaType]
                       inventoryItemDatabase: Database[InventoryItemSchemaType]
                       inventoryMonsterDatabase: Database[InventoryMonsterSchemaType]
                       userId: integer or None
                       visual: typed dictionary<"Visual",
                                                driver: Driver,
                                                toplevel: TopLevel,
                                                view: View or None,
                                                directory: str,
                                                splashes: dictionary[string, list[list[Rune]]]>>

Integer: INPUT1

function read_csv(file_name) → string
{ fungsi untuk read dari csv }
```

```
function custom_split(line: string, delimiter: string = ';') → string
{ fungsi untuk split string berdasarkan delimiter }

procedure show_mons_inv(None) → None
{ menampilkan monster yang dimiliki user }

procedure show_id_potion(None) → None
{ menampilkan potion yang ada di inventory user }

procedure mons_inv_detail(None) → None
{ menampilkan detail dari monster yang ada di inventory }

procedure show_potion_detail(gameState: GameState) → None
{ menampilkan detail dari potion yang ada di inventory }
```

```
ALGORITMA PROGRAM UTAMA
read_csv("item_inventory.csv")
read_csv("monster_inventory.csv")
read_csv("database_user.csv")
read_csv("monster.csv")
read_csv("user.csv")


mons_id ← read_csv("monster_inventory.csv")
mons_Lvl ← read_csv("monster_inventory.csv")
mons ← read_csv("monster.csv")
user_id ← read_csv("user.csv")
baca_oc ← read_csv("user.csv")
item_inv ← read_csv("item_inventory.csv")

input INPUT1
```

```
{ REALISASI FUNGSI/PROSEDUR }
function read_csv(file_name) → string
{ fungsi untuk read dari csv }
KAMUS LOKAL
    data : array
    row : string
ALGORITMA
   data ← []
   with open(file_name, 'r') as file
      FOR line IN file:
         row ← custom_split(line.strip())
         data.append(row)
   output data



function custom_split(line: string, delimiter: string = ';') → string
{ fungsi untuk split string berdasarkan delimiter }
```

```
KAMUS LOKAL
    parts: array
    current_part: string
    inside_quotes: boolean
ALGORITMA
   parts ← []
   current_part ← ''
   inside_quotes ← False
   FOR char IN line:
      if char = delimiter and not inside_quotes then
         parts.append(current_part)
         current_part ← ''
      else if char = '"' then
          inside_quotes ← not inside_quotes
      else
         current_part = current_part + char
   parts.append(current_part)
   output parts

procedure show_mons_inv(None) → None
{ menampilkan monster yang dimiliki user }
KAMUS LOKAL
      id ← integer
      m_type ← integer
      hp ← integer
      oc ← integer
      m_lvl ← integer
ALGORITMA
  i traversal [1, len(mons_id)-1]
      id ← mons_id[i][0]
      m_type ← mons[i][1]
      hp ← mons[i][4]
      oc ← baca_oc[i][4]
      m_lvl ← mons_Lvl[i][2]
      if gamestate_get_user_id() = id then
         output "============ INVENTORY LIST (User ID: {id}) ============"
         output " Jumlah O.W.C.A. Coin-mu sekarang {oc}."
         FOR monster IN len(mons_id):
            Output "{i}. Monster     (Name: {m_type}, Lvl: {m_lvl}, HP: {hp})"



procedure show_id_potion(None) → None
{ menampilkan potion yang ada di inventory user }
KAMUS LOKAL
    id_1 : integer
    type_1 : integer
    Qty: integer
ALGORITMA
   j traversal [length(mons_id) .. length(mons_id) + length(item_inv)]
```

```
        id_1 ← item_inv[j][0]
        type_1 ← item_inv[j][0]
        Qty ← item_inv[j][1]
        if gamestate_get_user_id() = id_1 then
            output "{j}. Potion       (Type: {type_1}, Qty: {Qty})"

procedure mons_inv_detail(None) → None
{ menampilkan detail dari monster yang ada di inventory }
KAMUS LOKAL
        Id: integer
        m_type: integer
        atk_power: integer
        def_power: integer
        hp: integer
        m_lvl: integer

ALGORITMA
    i traversal [1 .. length(mons_id)]
        id ← mons_id[i][0]
        m_type ← mons[i][1]
        atk_power ← mons[i][2]
        def_power ← mons[i][3]
        hp ← mons[i][4]
        m_lvl ← mons_Lvl[i][2]
        i traversal [1 .. length(mons_id)]
            if gamestate_get_user_id() = id then
                if INPUT1 = mons_id[i][2] then
                    output "Monster"
                    output "Name      : {m_type}"
                    output "ATK Power : {atk_power}"
                    output "DEF Power : {def_power}"
                    output "HP        : {hp}"
                    output "Level     : {m_lvl}"



procedure show_potion_detail(gameState: GameState) → None
{ menampilkan detail dari potion yang ada di inventory }
KAMUS LOKAL
     id_1: integer
     type_1: integer
     Qty: integer
ALGORITMA
    j traversal [len(mons_id) .. len(mons_id) + len(item_inv)]
        id_1 ← item_inv[j][0]
        type_1 ← item_inv[j][0]
        Qty ← item_inv[j][1]

        j traversal [len(mons_id)+1 .. len(mons_id) + len(item_inv)]
            if gamestate_get_user_id(gameState) = id_1 then
```

```
            if INPUT1 = j then
               output "Potion"
               output “Type      : {type_1}"
               output "Quantity  : {Qty}”
```

```
Program battleArena
{ berisi fungsi dan prosedur untuk fitur battle dan arena }
KAMUS
Type GameState: typed dictionary
                    <"GameState",
                       userDatabase: Database[UserSchemaType]
                       monsterDatabase: Database[MonsterSchemaType]
                        potionDatabase: Database[PotionSchemaType]
                       battleDatabase: Database[BattleSchemaType]
                       arenaDatabase: Database[ArenaSchemaType]
                       shopDatabase: Database[ShopSchemaType]
                       laboratoryDatabase: Database[LaboratorySchemaType]
                       inventoryItemDatabase: Database[InventoryItemSchemaType]
                       inventoryMonsterDatabase: Database[InventoryMonsterSchemaType]
                       userId: integer or None
                       visual: typed dictionary<"Visual",
                                                driver: Driver,
                                                toplevel: TopLevel,
                                                view: View or None,
                                                directory: str,
                                                splashes: dictionary[string, list[list[Rune]]]>>

type BattleSchemaType = named tuple("Battle", [
                                ("id": integer),
                                ("turn": integer (0, 1, atau 2)),
                                ("player1Id": integer),
                                ("player2Id": integer),
                                ("monster1Id": integer)
                                ("monster2Id": integer),
                                ("verdict": integer (-1, 0, 1, 2, 3, atau 4)), { -1 berarti battle
                            belum selesai, 0 berarti seri, 1 berarti player 1 menang, 2 berarti
                            player 2 menang, 3 berarti player 1 melarikan diri, 4 berari player
                            2 melarikan diri }
                                ("handler", str), # Handler or logic for current battle
                            ])

function battle_get(gameState : GameState, battleId : integer) → BattleSchemaType
{ menghasilkan data battle berdasarkan ID battle }

function battle_set(gameState : GameState, battleId : integer, modifier :
Union[BattleSchemaType, Callable[[BattleSchemaType], BattleSchemaType]]) →
BattleSchemaType
{ mengatur data battle berdasarkan ID battle dan pengubah }
```

```
function battle_new(gameState : GameState) → BattleSchemaType
{ membuat battle baru }

function battle_end_player_draw(gameState : GameState, battle : BattleSchemaType) →
BattleSchemaType
{ mengakhiri battle dengan hasil seri }

function _battle_end_player_won(gameState : GameState, battle : BattleSchemaType,
player12 : Literal[1, 2]) → BattleSchemaType
{ mengakhiri battle dengan salah satu pemain menang }

function battle_end_player_escaped(gameState : GameState, battle : BattleSchemaType,
player12 : Literal[1, 2]) → BattleSchemaType
{ mengakhiri battle dengan salah satu pemain melarikan diri }

function _battle_verdict_is_finished(verdict : integer) → boolean
{ memeriksa apakah battle telah selesai }

function _battle_verdict_is_player_draw(verdict : integer) → boolean
{ memeriksa apakah battle berakhir seri }

function battle_verdict_is_player_won(verdict : integer) → Optional[Literal[1, 2]]
{ memeriksa apakah pemain menang }

function _battle_verdict_is_player_escaped(verdict : integer) → Optional[Literal[1, 2]]
{ memeriksa apakah pemain melarikan diri }

function battle_action_attack(gameState : GameState, battle : BattleSchemaType) →
tuple[float, float]
{ melakukan attack dalam battle }

{ fungsi/prosedur arena }
function arena_get(gameState : GameState, arenaId : integer) → ArenaSchemaType
{ menghasilkan data arena berdasarkan ID arena }

function _arena_set(gameState : GameState, arenaId : integer, modifier :
Union[ArenaSchemaType, Callable[[ArenaSchemaType], ArenaSchemaType]]) →
ArenaSchemaType
{ mengatur data arena berdasarkan ID arena dan pengubah }

function _arena_new(gameState : GameState) → ArenaSchemaType
{ membuat arena baru }
```

```
ALGORITMA PROGRAM UTAMA
```

```
{ REALISASI FUNGSI/PROSEDUR }
{ fungsi/prosedur untuk battle }
function battle_get(gameState : GameState, battleId : integer) → BattleSchemaType
{ menghasilkan data battle berdasarkan ID battle }
KAMUS LOKAL
  battleDatabase : BattleDatabaseType
```

```
ALGORITMA
  battleDatabase ← gamestate_get_battle_database(gameState)
  output database_get_entry_at(battleDatabase, battleId)

function battle_set(gameState : GameState, battleId : integer, modifier :
Union[BattleSchemaType, Callable[[BattleSchemaType], BattleSchemaType]]) →
BattleSchemaType
{ mengatur data battle berdasarkan ID battle dan pengubah }
KAMUS LOKAL
  battleDatabase : BattleDatabaseType
  battle : BattleSchemaType
ALGORITMA
  battleDatabase ← gamestate_get_battle_database(gameState)
  battle ← modifier(database_get_entry_at(battleDatabase, battleId))
  if callable(modifier) else modifier
  database_set_entry_at(battleDatabase, battleId, battle)
  output battle

function battle_new(gameState : GameState) → BattleSchemaType
{ membuat battle baru }
KAMUS LOKAL
  battleDatabase : BattleDatabaseType
  battleId : integer
  battle : BattleSchemaType
ALGORITMA
  battleDatabase ← gamestate_get_battle_database(gameState)
  battleId ← database_get_entries_length(battleDatabase)
  battle ← BattleSchemaType(
    id = battleId,
    turn = null,
    player1Id = null,
    player2Id = null,
    monster1Id = null,
    monster2Id = null,
    verdict = null,
    handler = null
  )
  database_set_entry_at(battleDatabase, battleId, battle)
  output battle

function battle_end_player_draw(gameState : GameState, battle : BattleSchemaType) →
BattleSchemaType
{ mengakhiri battle dengan hasil seri }
KAMUS LOKAL

ALGORITMA
  output battle_set(gameState, battle.id, namedtuple_with(battle, verdict = 0))

function _battle_end_player_won(gameState : GameState, battle : BattleSchemaType,
player12 : Literal[1, 2]) → BattleSchemaType
```

```
{ mengakhiri battle dengan salah satu pemain menang }
KAMUS LOKAL

ALGORITMA
  output battle_set(gameState, battle.id, namedtuple_with(battle, verdict = if player12 = 1 then 1 else 2))

function battle_end_player_escaped(gameState : GameState, battle : BattleSchemaType, player12 : Literal[1, 2]) → BattleSchemaType
{ mengakhiri battle dengan salah satu pemain melarikan diri }
KAMUS LOKAL

ALGORITMA
  output battle_set(gameState, battle.id, namedtuple_with(battle, verdict = if player12 = 1 then 3 else 4))

function _battle_verdict_is_finished(verdict : integer) → boolean
{ memeriksa apakah battle telah selesai }
KAMUS LOKAL

ALGORITMA
  output verdict ≠-1

function _battle_verdict_is_player_draw(verdict : integer) → boolean
{ memeriksa apakah battle berakhir seri }
KAMUS LOKAL

ALGORITMA
  output verdict = 0

function battle_verdict_is_player_won(verdict : integer) → Optional[Literal[1, 2]]
{ memeriksa apakah salah satu pemain menang }
KAMUS LOKAL

ALGORITMA
  output if verdict = 1 then 1 else if verdict = 2 then 2 else null

function _battle_verdict_is_player_escaped(verdict : integer) → Optional[Literal[1, 2]]
{ memeriksa apakah salah satu pemain melarikan diri }
KAMUS LOKAL

ALGORITMA
  output if verdict = 3 then 1 else if verdict == 4 then 2 else null


function battle_action_attack(gameState : GameState, battle : BattleSchemaType) → tuple[float, float]
{ melakukan attack dalam battle }
KAMUS LOKAL
  selfMonsterId : integer
  opponentMonsterId : integer
```

```
  selfMonster : MonsterSchemaType
  opponentMonster : MonsterSchemaType
  selfMonsterAttack : float
  opponentMonsterDefense : float
  opponentMonsterHealth : float
ALGORITMA
  selfMonsterId ← if battle.turn == 1 then battle.monster1Id else if battle.turn == 2 then
battle.monster2Id else null
  opponentMonsterId ← if battle.turn == 1 then battle.monster2Id else if battle.turn == 2
then battle.monster1Id else null
  selfMonster ← inventory_monster_get(gameState, selfMonsterId)
  opponentMonster ← inventory_monster_get(gameState, opponentMonsterId)
  selfMonsterAttack ← inventory_monster_get_calculated_attack_power(gameState,
selfMonster)
  selfMonsterAttack ← selfMonsterAttack * 1.5 + selfMonsterAttack * 0.3 *
(gamestate_rand(gameState) * 2 - 1)
  opponentMonsterDefense ←
inventory_monster_get_calculated_defense_power(gameState, opponentMonster)
  opponentMonsterDefense ← opponentMonsterDefense * 0.2 + opponentMonsterDefense
* 0.2 * (gamestate_rand(gameState) * 2 - 1)
  selfMonsterAttack ← max(0, selfMonsterAttack - opponentMonsterDefense)
  opponentMonsterHealth ← max(0, opponentMonster.healthPoints - selfMonsterAttack)
  inventory_monster_set(gameState, opponentMonsterId,
namedtuple_with(opponentMonster, healthPoints = opponentMonsterHealth))
  output (selfMonsterAttack, opponentMonsterHealth)


{ fungsi/prosedur arena }
function arena_get(gameState : GameState, arenaId : integer) → ArenaSchemaType
{ menghasilkan data arena berdasarkan ID arena }
KAMUS LOKAL
  arenaDatabase : ArenaDatabaseType
ALGORITMA
  arenaDatabase ← gamestate_get_arena_database(gameState)
  output database_get_entry_at(arenaDatabase, arenaId)


function _arena_set(gameState : GameState, arenaId : integer, modifier :
Union[ArenaSchemaType, Callable[[ArenaSchemaType], ArenaSchemaType]]) →
ArenaSchemaType
{ mengatur data arena berdasarkan ID arena dan pengubah }
KAMUS LOKAL
  arenaDatabase : ArenaDatabaseType
  arena : ArenaSchemaType
ALGORITMA
  arenaDatabase ← gamestate_get_arena_database(gameState)
  arena ← modifier(database_get_entry_at(arenaDatabase, arenaId)) if callable(modifier)
else modifier
  database_set_entry_at(arenaDatabase, arenaId, arena)
  output arena
```

```
function _arena_new(gameState : GameState) → ArenaSchemaType
{ membuat arena baru }
KAMUS LOKAL
  arenaDatabase : ArenaDatabaseType
  arenaId : integer
  arena : ArenaSchemaType
ALGORITMA
  arenaDatabase ← gamestate_get_arena_database(gameState)
  arenaId ← database_get_entries_length(arenaDatabase)
  arena ← ArenaSchemaType(
                                    id = arenaId,
                                    playerId = null,
                                    battleIds = null,
                                    handler = null
                                  )
  database_set_entry_at(arenaDatabase, arenaId, arena)
  output arena
```

```
Program shop
{ berisi fungsi/prosedur untuk fitur F10 }
KAMUS
Type GameState: typed dictionary
                    <"GameState",
                       userDatabase: Database[UserSchemaType]
                       monsterDatabase: Database[MonsterSchemaType]
                        potionDatabase: Database[PotionSchemaType]
                       battleDatabase: Database[BattleSchemaType]
                       arenaDatabase: Database[ArenaSchemaType]
                       shopDatabase: Database[ShopSchemaType]
                       laboratoryDatabase: Database[LaboratorySchemaType]
                       inventoryItemDatabase: Database[InventoryItemSchemaType]
                       inventoryMonsterDatabase: Database[InventoryMonsterSchemaType]
                       userId: integer or None
                       visual: typed dictionary<"Visual",
                                                driver: Driver,
                                                toplevel: TopLevel,
                                                view: View or None,
                                                directory: str,
                                                splashes: dictionary[string, list[list[Rune]]]>>

integer: x, user_id, oc,
string: item_Shop, mons, item_inv, mons_shop, mons_inv, baca_oc

function SHOW_ID_MONSTER(None) → None
{ memberikan informasi mengenai monster yang dijual dalam shop }
```

```
function SHOW_ID_POTION(None) → None
{ memberikan informasi mengenai potion yang dijual dalam shop }

procedure BELI_MONSTER(gameState: GameState) → None
{ prosedur untuk membeli monster }

procedure BELI_POTION(gameState: GameState) → None
{ prosedur untuk membeli potion }
```

```
ALGORITMA PROGRAM UTAMA
output("<<<SHOP>>>")
output("Irassahaimase! Selamat datang di SHOP!!")
read_csv("item_inventory.csv")
read_csv("item_shop.csv")
read_csv("user.csv")
read_csv("monster_shop.csv")
read_csv("monster.csv")

item_Shop ← read_csv("item_shop.csv")
mons ← read_csv("monster.csv")
item_inv ← read_csv("item_inventory.csv")
mons_shop ← read_csv("monster_shop.csv")
mons_inv ← read_csv("monster_inventory.csv")
baca_oc ← read_csv("user.csv")
x ← 0

user_id ← gamestate_get_user_id(gameState)
user_database ← gamestate_get_user_database(gameState)
user_entry ← database_get_entries_at(user_database, user_id)
oc ← user_entry.money

while x = 0 do
   input("Pilih aksi (lihat/beli/keluar): ", aksi)
   if aksi = "keluar" then
      output("Mr. Yanto bilang makasih, belanja lagi ya nanti :)")
      x ← x + 1
   else if aksi = "lihat" then
      input("Mau lihat apa (monster/potion): ", aksi_2)
      if aksi_2 = "monster" then
         output("ID |Type |ATK Power |DEF Power |HP |Stok |Harga")
         SHOW_ID_MONSTER
      else if aksi_2 = "potion" then
         output("ID |Type |Stok |Harga")
         SHOW_ID_POTION
   else if aksi = "beli" then
      output("jumlah O.W.C.A coin-mu sekarang ", oc)
      input("Mau beli apa? (monster/potion): ", aksi_3)
      if aksi_3 = "monster" or aksi_3 = "Monster" or aksi_3 = "MONSTER" then
         input("Masukkan id monster: ", aksi_beli_monster)
         BELI_MONSTER
```

```
        else if aksi_3 = "potion" or aksi_3 = "Potion" or aksi_3 = "POTION" then
            input("Masukkan id potion: ", aksi_beli_potion)
            input("Masukkan jumlah potion: ", jumlah_beli_potion)
            BELI_POTION
```

```
{ REALISASI FUNGSI/PROSEDUR }
function SHOW_ID_MONSTER(None) → None
{ memberikan informasi mengenai monster yang dijual dalam shop }
    i traversal [1 ... length(mons_shop) - 1]
        id ← mons_shop[i][0]
        type ← mons[i][1]
        atk ← mons[i][2]
        defence ← mons[i][3]
        hp ← mons[i][4]
        stok ← mons_shop[i][1]
        price ← mons_shop[i][2]
        output(id, " |", type, " |", atk, " |", defence, " |", hp, " |", stok, " |", price)

function SHOW_ID_POTION(None) → None
{ memberikan informasi mengenai potion yang dijual dalam shop }
    j traversal [1 ... length(item_Shop) - 1]
        id_1 ← j
        type_1 ← item_Shop[j][0]
        stok_1 ← item_Shop[j][1]
        harga_1 ← item_Shop[j][2]
        output(id_1, " |", type_1, " potion |", stok_1, " |", harga_1)

procedure BELI_MONSTER(gameState: GameState) → None
{ prosedur untuk membeli monster }
    i traversal [1 ... length(mons_shop) - 1]
        id ← mons_shop[i][0]
        type ← mons[i][1]
        type_inv_id ← mons_inv[i][1]
        stok ← mons_shop[i][1]
        price ← mons_shop[i][2]
        oc ← user_entry.money
        if aksi_beli_monster = id then
            oc ← integer(oc)
            price ← integer(price)
            stok ← integer(stok)
            if aksi_beli_monster = type_inv_id then
                output("Monster sudah ada di inventory kamu :), transaksi dibatalkan")
            else
                if stok >= 1 then
                    if oc >= price then
                        output("kamu berhasil membeli ", type, ", dan ", type, " telah disimpan di
inventory.")
                        output("Sisa O.C kamu ", oc - price)
                        oc ← oc - price
                        oc ← str(oc)
```

```
                baca_oc[pilih_id][4] ← oc
                stok ← stok - 1
                stok ← str(stok)
                mons_shop[i][1] ← stok
            else if oc < price then
                output("O.C kamu kurang :(")
        else
            output("Stok monster habis")

procedure BELI_POTION(gameState: GameState) → None
{ prosedur untuk membeli potion }
  j traversal [1 ... length(item_Shop) - 1]
    id_1 ← j
    type_1 ← item_Shop[j][0]
    stok_1 ← item_Shop[j][1]
    harga_1 ← item_Shop[j][2]
    oc ← user_entry.money
    if aksi_beli_potion = id_1 then
      oc ← integer(oc)
      harga_1 ← integer(harga_1)
      stok_1 ← integer(stok_1)
      jumlah ← integer(jumlah_beli_potion * harga_1)
      if stok_1 >= 1 then
        if oc >= harga_1 then
          output("kamu berhasil membeli ", type_1)
          output("Sisa O.C kamu ", oc - jumlah, ", sebanyak ", jumlah_beli_potion)
          oc ← oc - jumlah
          oc ← str(oc)
          baca_oc[pilih_id][4] ← oc
          stok_1 ← stok_1 - 1
          stok_1 ← str(stok_1)
          item_Shop[j][1] ← stok_1
        else if oc = harga_1 then
          input("O.C kamu pas-pasan untuk beli potion... mau lanjut beli? (Y/N): ",
beli_ga)
          if beli_ga = "Y" or beli_ga = "y" then
            output("kamu berhasil membeli ", type_1)
            output("Sisa O.C kamu ", oc - jumlah, ", sebanyak ", jumlah_beli_potion)
            oc ← oc - jumlah
            oc ← str(oc)
            baca_oc[pilih_id][4] ← oc
            stok_1 ← stok_1 - 1
            stok_1 ← str(stok_1)
            item_Shop[j][1] ← stok_1
          else if beli_ga = "N" or beli_ga = "n" then
            output("Okay transaksi dibatalkan, selamat menabung!!")
        else if oc < harga_1 then
          output("O.C kamu kurang :(")
      else
        output("Stok potion habis")
```

```
Program laboratory
{ berisi fungsi/prosedur untuk fitur laboratory }
```

```
KAMUS
Type GameState: typed dictionary
                    <"GameState",
                       userDatabase: Database[UserSchemaType]
                       monsterDatabase: Database[MonsterSchemaType]
                        potionDatabase: Database[PotionSchemaType]
                       battleDatabase: Database[BattleSchemaType]
                       arenaDatabase: Database[ArenaSchemaType]
                       shopDatabase: Database[ShopSchemaType]
                       laboratoryDatabase: Database[LaboratorySchemaType]
                       inventoryItemDatabase: Database[InventoryItemSchemaType]
                       inventoryMonsterDatabase: Database[InventoryMonsterSchemaType]
                       userId: integer or None
                       visual: typed dictionary<"Visual",
                                                driver: Driver,
                                                toplevel: TopLevel,
                                                view: View or None,
                                                directory: str,
                                                splashes: dictionary[string, list[list[Rune]]]>>

string: mons, mons_inv, user_database, user_entry
integer: user_id, baca_oc
procedure Laboratory (user_id: integer, tabel_Mons_inv, tabel_Mons, baca_oc: string) →
None
{ Menampilkan daftar monster dan harga upgrade serta melakukan upgrade level monster }
```

```
ALGORITMA PROGRAM UTAMA
mons ← read_csv("monster.csv")
mons_inv ← read_csv("monster_inventory.csv")

user_id ← gamestate_get_user_id(gameState)
user_database ← gamestate_get_user_database(gameState)
user_entry ← database_get_entries_at(user_database, user_id)
baca_oc ← user_entry.money
```

```
{ REALISASI FUNGSI/PROSEDUR }
procedure Laboratory (user_id: integer, tabel_Mons_inv, tabel_Mons, baca_oc: integer) →
None
{ Menampilkan daftar monster dan harga upgrade serta melakukan upgrade level monster }

KAMUS LOKAL
   level, id, oc, levels, levelb, ocs, sisa_ocs: integer
   harga: array[0..4] of integer = [0, 300, 500, 800, 1000]
   choice, j: integer
   lanjut_upgrade: char
```

```
ALGORITMA
   output("============ MONSTER LIST ============")
   j ← 1
   i traversal [0... length(tabel_Mons_inv) - 1]
      if tabel_Mons_inv[i][0] = user_id then
         level ← tabel_Mons_inv[i][2]
         id ← tabel_Mons_inv[i][1]
         k traversal [0... length(tabel_Mons) - 1]
            if id = tabel_Mons[k][0] then
               output(j, ". ", tabel_Mons[k][1], " (level: ", level, ")")
               j ← j + 1

   output("============ UPGRADE PRICE ============")
   output("1. Level 1 -> Level 2: 300 OC")
   output("2. Level 2 -> Level 3: 500 OC")
   output("3. Level 3 -> Level 4: 800 OC")
   output("4. Level 4 -> Level 5: 1000 OC")

   i traversal [0... length(baca_oc) - 1] do
      if baca_oc[i][0] = user_id then
         oc ← baca_oc[i][4]
         output(oc)

   input("Pilih monster: ", choice)
   j ← 0
   for i traversal [0... length(tabel_Mons_inv) - 1] do
      if tabel_Mons_inv[i][0] = user_id then
         level ← tabel_Mons_inv[i][2]
         id ← tabel_Mons_inv[i][1]
         for k traversal [0... length(tabel_Mons) - 1] do
            if id = tabel_Mons[k][0] then
               j ← j + 1
               if choice = j then
                  levels ← integer(level)
                  levelb ← integer(level)
                  ocs ← integer(oc)
                  output(ocs)
                  if levels >= 5 then
                     output("Maaf, monster yang Anda pilih sudah memiliki level
maksimum")
                  else
                     output(tabel_Mons[k][1], " akan di-upgrade ke level ", levels + 1)
                     output("Harga untuk melakukan upgrade ", tabel_Mons[k][1], " adalah ",
harga[levels], " OC")
                     input("Lanjutkan upgrade (Y/N): ", lanjut_upgrade)
                     if lanjut_upgrade = "Y" or lanjut_upgrade = "y" then
                        if ocs >= harga[levels] then
                           output("Selamat, ", tabel_Mons[k][1], " berhasil di-upgrade ke level
", levels + 1)
                           sisa_ocs ← ocs - harga[levelb]
```

```
                        output("sisa OWCA kamu :", sisa_ocs)
                        levels ← levels + 1
                        baca_oc[i][4] ← str(sisa_ocs)
                        tabel_Mons_inv[i][2] ← str(levels)
                    else
                        output("OC Anda kurang, maka monster ", tabel_Mons[k][1], " tidak
dapat di-upgrade :(")
                else if lanjut_upgrade = "N" or lanjut_upgrade = "n" then
                    output("Monster ", tabel_Mons[k][1], " tidak jadi di-upgrade")
                else
                    output("Mohon maaf command yang anda masukan salah")
```

```
Program database
{ berisi fungsi/prosedur untuk save, load, dan exit dari game }
KAMUS
type DatabaseSchemaType: Named Tuple

function _database_schema_read_handle(schema:
DatabaseSchema[DatabaseSchemaType], handle: str) →  list[_DatabaseSchemaType]
{ membaca handle dari skema database }

function zipData(entry: list[str]) → dictionary[string, string]
{ mengubah list of string menjadi dictionary }

procedure _database_schema_write_handle(schema:
DatabaseSchema[_DatabaseSchemaType], handle: str, entries:
list[_DatabaseSchemaType]) → None
{ menulis handle ke  skema database }

function unzipData(entry: dict[str, str]) → list[string]
{ mengubah dictionary dengan key string dan value string menjadi list of string }

procedure database_load(database: Database[_DatabaseSchemaType]) →  None
{ load data program dari folder save }

procedure database_save(database: Database[_DatabaseSchemaType]) →  None
{ save data program dari folder save }

procedure exit(None) →  None
{ prosedur untuk exit dari game }
ALGORITMA PROGRAM UTAMA

{ REALISASI FUNGSI/PROSEDUR }
function _database_schema_read_handle(schema:
DatabaseSchema[DatabaseSchemaType], handle: str) →  list[_DatabaseSchemaType]
{ membaca handle dari skema database }
KAMUS LOKAL
    format: “format”
```

```
    properties: “properties”
ALGORITMA
   format ← schema["format"]
   properties ← schema["properties"]
   if format = "csv" then
      function zipData(entry: list[str]) → dictionary[string, string]
      { mengubah list of string menjadi dictionary }
         KAMUS LOKAL
            result: dictionary
         ALGORITMA
         result ← dict()
         i traversal [0 .. length(properties)-1]
            result[_database_property_get_name(properties[i])] ← entry[i]
         output result
      output array_map(csv_read_from_file(handle), lambda e, *_:
_database_schema_decode_entry(schema, zipData(e)))

procedure _database_schema_write_handle(schema:
DatabaseSchema[_DatabaseSchemaType], handle: str, entries:
list[_DatabaseSchemaType]) → None
{ menulis handle ke  skema database }
KAMUS LOKAL
    format: “format”
    properties: “properties”
ALGORITMA
   format ← schema["format"]
   properties ← schema["properties"]
   if format = "csv" then
      function unzipData(entry: dict[str, str]) → list[string]
      { mengubah dictionary dengan key string dan value string menjadi list of string }
         KAMUS LOKAL

         ALGORITMA
         output array_map(properties, lambda p, *_:
entry[_database_property_get_name(p)])
      csv_write_to_file(handle, array_map(entries, lambda e, *_:
unzipData(_database_schema_encode_entry(schema, e))))
      output

procedure database_load(database: Database[_DatabaseSchemaType]) →  None
{ load data program dari folder save }
KAMUS LOKAL
    handle: “handle”
    schema: “schema”
    database[“entries”]: array
ALGORITMA
   handle ← database["handle"]
   schema ← database["schema"]
   database["entries"] ← database_schema_read_handle(schema, handle)
```

```
procedure database_save(database: Database[_DatabaseSchemaType]) → None
{ save data program dari folder save }
KAMUS LOKAL
    handle: “handle”
    schema: “schema”
ALGORITMA
   handle ← database["handle"]
   schema ← database["schema"]
   database_schema_write_handle(schema, handle, database["entries"])

procedure exit(None) → None
{ prosedur untuk exit dari game }
KAMUS LOKAL
    konfirmasiSave: string
ALGORITMA
     input konfirmasiSave
     if konfirmasiSave = "y" or konfirmasiSave = "Y" then
        database_save()
        SystemExit()
     else if konfirmasiSave = "n" or konfirmasiSave = “N” then
        SystemExit()
```

F-12 Shop Management

output(“Selamat datang”, user)

```
KAMUS
aksi, lihat, ubah, nambah, hapus : string
monsters_complete : array of integer and string = [
[“ID, “Type”, “ATK Power”, “DEF Power”, “HP”, “Stok”, “Harga”],
[1, “Pikachow”, 125,10,600,10,500],
[2, “Bulbu”, 50,50,1200,4,700],
[3, “Zeze”, 300,10,100,3,300],
[4, “Zuko”, 100,25,800,8,550],
[5,”Chacha”,80,30,700,7,600],
]
item_shops : array of integer and string = [
[“ID”, “Type”, “Stok”, “Harga”],
[1, “Strength Potion”, 10, 50],
[2, “Resilience Potion”,5,30],
[3, “Healing Potion”, 3, 20],
]
monsters : array of integer and string = [
[“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],
]
```

```
id, stokAwal, harga, stokBaru, hargaBaru : integer

ALGORITMA
output("Pilih aksi (lihat/tambah/ubah/hapus/keluar) : ", input(aksi)

depend on (aksi)
aksi = "lihat" : output("Mau lihat apa? (monster/potion): "), input(lihat)
aksi = "nambah" : output("Mau nambahin apa? (monster/potion): ", input(nambah)
aksi = "ubah" : output("Mau ngubah apa (monster/potion): ?", input(ubah)
aksi = "hapus" : output("Mau hapu apa? (monster/potion) : ", input(hapus)
else (aksi = "keluar") :
        output ("Dadah", user , " , sampai jumpa di lain waktu!"

depend on (aksi, lihat,nambah,ubah)
        (aksi = "lihat") and(lihat = "monster") :
        for row in monsters_complete:
                output("|".join(map(str,row)))
        output("Pilih aksi (lihat/tambah/ubah/hapus/keluar) : ", input(aksi)

        (aksi = "lihat") and (lihat = "potion") :
             for row in item_shops
                output("|".join(map(str,row)))
        output("Pilih aksi (lihat/tambah/ubah/hapus/keluar) : ", input(aksi)

        (aksi = "tambah") and (nambah = "monster") :
        for row in monsters
            output("|".join(map(str,row)))
        output("Masukkan id monster: "), input(id)
        output("Masukkan stok awal : "), input(stokAwal)
         output("Masukkan harga: "), input(harga)

         depend on (id)
        id = 1 :
              new0 ← [id, stokAwal, harga]
              monsters[0] ← new0
              for row in monsters:
                 output("Pikachow berhasil ditambahkan ke dalam shop")
         id = 2 :
                 new1 ← [id, stokAwal, harga]
                 monsters[1] ← new1
                 for row in monsters:
                   output("Bulbu berhasil ditambahkan ke dalam shop")
          id = 3 :
                 new2  ← [id, stokAwal, harga]
                  monsters[2] ← new2
                  For row in monsters:
                    output("Zeze berhasil ditambahkan ke dalam shop")
           id = 4 :
                  new3  ← [id, stokAwal, harga]
                  monsters[3] ← new3
                  for row in monsters:
                       output("Zuko berhasil ditambahkan ke dalam shop")
        else {id = 5}
```

```
                    new3 ← [id, stokAwal, harga]
                    monsters[4] ← new4
                     for row in monsters:
                       output("Zuko berhasil ditambahkan ke dalam shop")
            output("Pilih aksi (lihat/tambah/ubah/hapus/keluar) : ", input(aksi)


(aksi = "tambah") and (nambah = "potion") :
     for row in item_shops:
       output("|".join(map(str,row))
       output("Masukkan id: "), input(id)
       output("Masukkan stok awal : "), input(stokAwal)
       output("Masukkan harga: "), input(harga)
       depend on (id)
          id = 1 :
                    New0 ← [id, stokAwal, harga]
                     item_shops[0] ← new0
                    for row in item_shops:
                       output("Strength Potion berhasil ditambahkan ke dalam shop")
           id = 2 :
                    new1  ← [id, stokAwal, harga]
                    monsters[1] ← new1
                     for row in item_shops:
                        output("Resilience Potion berhasil ditambahkan ke dalam shop")
          else {id = 3}
                   new2 ← [id, stokAwal, harga]
                    monsters[2] ← new2
                    for row in item_shops:
                        output("Healing Potion berhasil ditambahkan ke dalam shop")
           output("Pilih aksi (lihat/tambah/ubah/hapus/keluar) : ", input(aksi)

(aksi = "ubah") and (ubah = "monster")
 for row in monsters_complete
     output("|".join(map(str,row)))
     output("Masukkan id monster: "), input(id)
     output("Masukkan stok baru : "), input(stokBaru)
     output("Masukkan harga baru2: "), input(hargaBaru)
        depend on (id)
        id = 1 :
                    baru0 = [id, stokBaru, hargaBaru]
                    monsters_complete[0] = baru0
                      for row in monsters_complete:
                       depend on (hargaBaru)
                       hargaBaru != "" :
                              output("Pikachow telah berhasil diubah dengan stok baru sejumlah" ,
                    stokBaru,"!")
                     hargaBaru = hargaBaru :
                              output("Pikachow telah berhasil diubah dengan stok baru sejumlah ",
                    stokBaru, "dan dengan harga", hargaBaru "!")
          id = 2 :
                    baru1 = [id, stokBaru, hargaBaru]
                    monsters_complete[1] = baru1
                    for row in monsters_complete:
                    depend on (hargaBaru)
```

```
                    hargaBaru != ‘’” :
                            output(“Bulbu telah berhasil diubah dengan stok baru
                    sejumlah” ,stokBaru,”!”)
                    hargaBaru = hargaBaru :
                            output(“Bulbu telah berhasil diubah dengan stok baru sejumlah “,
                    stokBaru, “dan dengan harga”, hargaBaru “!”)
        id = 3 :
                    baru2 = [id, stokBaru, hargaBaru]
                    monsters_complete[2] = baru2
                    for row in monsters_complete:
                    depend on (hargaBaru)
                    hargaBaru != ‘’” :
                            output(“Zeze telah berhasil diubah dengan stok baru
                    sejumlah” ,stokBaru,”!”)
                    hargaBaru = hargaBaru :
                            output(“Zeze telah berhasil diubah dengan stok baru sejumlah “, stokBaru,
                    “dan dengan harga”, hargaBaru “!”)
        id = 4 :
                    baru3 = [id, stokBaru, hargaBaru]
                    monsters_complete[3] = baru3
                     for row in monsters_complete:
                      depend on (hargaBaru)
                          hargaBaru != ‘’” :
                            output(“Zuko telah berhasil diubah dengan stok baru
                    sejumlah” ,stokBaru,”!”)
                          hargaBaru == hargaBaru :
                            output(“Zuko telah berhasil diubah dengan stok baru sejumlah “, stokBaru,
                    “dan dengan harga”, hargaBaru “!”)

(aksi ==  “ubah”) and (ubah == “potion”):
       for row in item_shops:
         output(“|”.join(map(str,row)))
      output(“Masukkan id potion : “), input(id)
      output(“Masukkan stok baru : “ ), input(stokBaru)
     output(“Masukkan harga baru : “), input(hargaBaru)
     depend on (id)
       id = 1 :
        Baru0 = [id, stokBaru, hargaBaru]
        item_shops[0] = Baru0
        for row in item_shops:
        depend on (stokBaru)
            stokBaru = “” :
                output(“Strength Potion telah berhasil diubah dengan stok baru
sejumlah” ,stokBaru,”!”)
             stokBaru == stokBaru :
                            output(“Strength Potion telah berhasil diubah dengan stok baru sejumlah “,
                    stokBaru, “dan dengan harga”, hargaBaru “!”)
        id = 2 :
         Baru1 = [id, stokBaru, hargaBaru]
         item_shops[1] = Baru1
         for row in item_shops:
           depend on (stokBaru)
                    stokBaru != ‘’” :
```

```
                    output("Resilience Potion telah berhasil diubah dengan stok baru
                sejumlah" ,stokBaru,"!")
                stokBaru = stokBaru :
                    output("Resilience Potion telah berhasil diubah dengan stok baru sejumlah
                ", stokBaru, "dan dengan harga", hargaBaru "!")
        else {id = 3}
          Baru2 = [id, stokBaru, hargaBaru]
          item_shops[2] = Baru2
          for row in item_shops:
              depend on (stokBaru)
                stokBaru != '"" :
                    output("Healing Potion telah berhasil diubah dengan stok baru
                sejumlah" ,stokBaru,"!")
              stokBaru = stokBaru :
                    output("Healing Potion telah berhasil diubah dengan stok baru sejumlah ",
                stokBaru, "dan dengan harga", hargaBaru "!")

(aksi = "hapus") and (hapus = "monster"):
for row in monster
   output("|".join(map(str,row)))
   output("Masukkan id potion: ")
     depend on (id)
      id = 1  :
      output("Apakah anda yakin ingin menghapus Pikachow dari shop (y/n): ")
     if (yakin = "y") then
        monsters_complete ← monster_complete
         del monsters_complete[0]:
         output("Pikachow berhasil dihapus dari shop!")
      id = 2 :
        output("Apakah anda yakin ingin menghapus Bulbu dari shop (y/n): ")
         if (yakin = "y") then
            monsters_complete ← monster_complete
             del monsters_complete[1]:
             output("Bulbu berhasil dihapus dari shop!")
     id = 3 :
        output("Apakah anda yakin ingin menghapus Zeze dari shop (y/n): ")
       if (yakin = "y") then
           monsters_complete ← monster_complete
          del monsters_complete[2]:
          output("Zeze berhasil dihapus dari shop!")
      id = 4 :
      output("Apakah anda yakin ingin menghapus Zuko dari shop (y/n): ")
      if (yakin = "y") then
        monsters_complete ← monster_complete
        del monsters_complete[3]:
        output("Zuko berhasil dihapus dari shop!")
     id = 5 :
     output("Apakah anda yakin ingin menghapus Chaca dari shop (y/n): ")
     if (yakin == "y") then
        monsters_complete ← monster_complete
       del monsters_complete[4]:
       output("Chaca berhasil dihapus dari shop!")
(aksi = "hapus") and (hapus = "potion")
for row in item_shops :
```

```
  output("|".join(map(str,row)))
  output("Masukkan id potion: "), input(id)
  depend on (id)
  id = 1  :
  output("Apakah anda yakin ingin menghapus Strength Potion dari shop (y/n): ")
  if (yakin = "y") then
    item_shops ← item_shops
    del item_shops[0] :
      output("Strength Potion berhasil dihapus dari shop!")
 id = 2 :
  output("Apakah anda yakin ingin menghapus Resilience Potion dari shop (y/n): ")
   if (yakin = "y") then
     item_shops ← item_shops
    del item_shops[1] :
       output("Resilience Potion berhasil dihapus dari shop!")
id = 3
output("Apakah anda yakin ingin menghapus Healing Potion dari shop (y/n): ")
if (yakin == "y") then
    item_shops ← item_shops
    del item_shops[2] :
       output("Healing Potion berhasil dihapus dari shop!")

(aksi = "keluar"):
output("Dadah", user , "sampai jumpa di lain waktu!")
```

F-13 Monster Management

```
output("SELAMAT DATANG DI DATABASE PARA MONSTER")
output("1. Tampilkan semua monster")
output("2. Tambah Monster baru")
```

```
KAMUS LOKAL
aksi, no, ATKpower, Def, HP : integer
masukkan : string
monsters : array of integer and string = [
["ID, "Type", "ATK Power", "DEF Power", "HP", "Stok", "Harga"],
[1, "Pikachow", 125,10,600,10,500],
[2, "Bulbu", 50,50,1200,4,700],
[3, "Zeze", 300,10,100,3,300],
[4, "Zuko", 100,25,800,8,550],
[5,"Chacha",80,30,700,7,600],
```

```
ALGORITMA
output("Pilih aksi: "), input(aksi)
if (aksi = 1) then
 for row in data
          output("|".join(maps(str,row)))
else {aksi = 2}
           no ←←← 5
output("Memulai pembuatan Monster baru")
output("Masukkan Type / Nama : "), input(type)
```

```
if (type = “Pikachow”) or (type = “Bulbu”) or (type = “Zeze”) or (type   = “Zuko”) or (type =
“Chacha”)
]output(“Nama sudah terdaftar, coba lagi!”)


iterate
no <- no + 1
stop output(“Masukkan TK Power : “), input(ATKpower)
      except:
       output(“Masukkan input berupa Integer, coba lagi!”)


output(“Masukkan DEF Power: “), input(Def)
while 0<= def <= 50 do
            stop output(“Masukkan HP  : “), input(HP)
            except:
               output(“DEF Power harus bernilai 0-50, coba lagi!”

output(“Monster baru berhasil dibuat!”)
output(“Type : “, type)
output(“ATK Power : “, ATKpower)
output(“DEF Power: “, Def)
output(“HP : “, HP)
output(“Tambahkan Monster ke database (Y/N) : “)
if (masukkan  “Y”) then
    output(“ID | Type | ATK Power | DEF Power | HP “)
    new_monster ←← [no, type, ATKpower, Def, HP]
    new_data ←← data_monster + [new_monster]
    new_data ←← data_monster
    for row in data_monster :
output(“|”.join(map(str,row)))
```

Program Primordials

```
Program primordials
{ berisi fungsi dan prosedur primordial untuk memanipulasi string dan array}
```

```
KAMUS
{ FUNGSI PRIMORDIAL STRING }
function _string_concat (strings : sequence of string) → string
 { menghasilkan hasil konkatenasi semua string dalam strings }

function _string_slice (string : string, start : integer, end : integer = None) → string
{ menghasilkan substring dari string dimulai dari indeks start hingga sebelum indeks end }
```

function _string_substring (string : string, start : integer, end : integer = None) → string
{ menghasilkan substring dari string dimulai dari indeks start hingga sebelum indeks end dengan menggunakan _string_slice }

function _string_code_point_at (string : string, i : integer) → integer
{ menghasilkan kode titik untuk karakter pada indeks i dalam string }

function _string_index_of (string : string, search : string, position : integer = 0) → integer
{ menghasilkan indeks pertama dari kemunculan substring search dalam string mulai dari posisi tertentu }

function _string_last_index_of (string : string, search : string, position : integer = None) → integer
{ menghasilkan indeks terakhir dari kemunculan substring search dalam string mulai dari posisi tertentu }

function _string_includes (string : string, search : string) → boolean
{ menghasilkan True jika search ditemukan dalam string, False jika tidak }

function _string_starts_with (string : string, search : string) → boolean
{ menghasilkan True jika string dimulai dengan substring search, False jika tidak }

function _string_ends_with (string : string, search : string) → boolean
{ menghasilkan True jika string diakhiri dengan substring search, False jika tidak }

function _string_pad_start (string : string, length : integer, padding : string) → string
{ menambahkan padding di awal string sehingga panjangnya menjadi length }

function _string_pad_end (string : string, length : integer, padding : string) → string
{ menambahkan padding di akhir string sehingga panjangnya menjadi length }

function _string_repeat (string : string, count : integer) → string
{ menghasilkan string yang diulang sebanyak count kali }

function _string_to_upper_case (string : string) → string
{ menghasilkan string dengan semua huruf diubah menjadi huruf besar }

function _string_to_lower_case (string : string) → string
{ menghasilkan string dengan semua huruf diubah menjadi huruf kecil }

function _string_trim_end (string: String) → String
{ Menghapus spasi dan pemisah baris dari akhir string }

function _string_trim_start (string: String) → String
{ Menghapus spasi dan pemisah baris dari awal string }

function _string_trim (string: String) → String
{ Menghapus spasi dan pemisah baris dari kedua ujung string }

```
function _string_replace (string: String, search: String, replacement: String/Function) → String
{ Mengganti kemunculan pertama dari search dengan replacement dalam string  }

function _string_replace_all (string: String, search: String, replacement: String/Function) → String
{ Mengganti semua kemunculan dari search dengan replacement dalam string }

function _string_split (string: String, separator: String) → List<String>
{ Membagi string dengan separator dan mengembalikan daftar substring }

{ FUNGSI/PROSEDUR PRIMORDIAL ARRAY }
function _array_push (array: List, *elements) → Integer
{ Menambahkan satu atau lebih elemen ke akhir array dan mengembalikan panjang baru array }

function _array_pop (array: List) → Any
{ Menghapus elemen terakhir dari array dan mengembalikan elemen tersebut }

function _array_unshift (array: List, *elements) → Integer
{ Menambahkan satu atau lebih elemen ke awal array dan mengembalikan panjang baru array }

function _array_shift (array: List) → Any
{ Menghapus elemen pertama dari array dan mengembalikan elemen tersebut }

function _array_concat (*arrays: List) → List
{ Menggabungkan dua atau lebih array dan mengembalikan array baru }

function _array_slice (array: List, start: Integer = 0, end: Integer = None) → List
{ Mengembalikan salinan dangkal dari sebagian array menjadi objek array baru yang dipilih dari start hingga end }

function _array_splice (array: List, start: Integer, deleteCount: Integer = None, *elements) → List
{ Mengubah isi array dengan menghapus atau mengganti elemen yang ada dan/atau menambahkan elemen baru di tempatnya }

function _array_copy_within (array: List, target: Integer, start: Integer, end: Integer = None) → List
{ Menyalin sebagian array secara dangkal ke lokasi lain dalam array yang sama dan mengembalikannya tanpa mengubah panjangnya }

function _array_every (array: List, callback: Function) → Boolean
{ Menguji apakah semua elemen dalam array lulus uji yang diimplementasikan oleh fungsi yang disediakan }
```

function _array_some (array: List, callback: Function) → Boolean
{ Menguji apakah setidaknya satu elemen dalam array lulus uji yang diimplementasikan oleh fungsi yang disediakan }

function _array_index_of (array: List[_T], element: _T) → Integer
{ Mengembalikan indeks pertama di mana elemen ditemukan dalam array, atau -1 jika tidak ditemukan }

function _array_last_index_of (array: List[_T], element: _T) → Integer
{ Mengembalikan indeks terakhir di mana elemen ditemukan dalam array, atau -1 jika tidak ditemukan }

function _array_includes (array: List[_T], element: _T) → Boolean
{ Memeriksa apakah array berisi elemen yang ditentukan }

function _array_find (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] → Boolean) → _T
{ Mengembalikan elemen pertama dalam array yang lulus uji yang diberikan oleh fungsi callback }

function _array_find_index (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] → Boolean) → Integer
{ Mengembalikan indeks elemen pertama dalam array yang lulus uji yang diberikan oleh fungsi callback }

function _array_find_last (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] → Boolean) → _T
{ Mengembalikan elemen terakhir dalam array yang lulus uji yang diberikan oleh fungsi callback }

function _array_find_last_index (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] → Boolean) → Integer
{ Mengembalikan indeks elemen terakhir dalam array yang lulus uji yang diberikan oleh fungsi callback }

function _array_filter (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] → Boolean) → List[_T]
{ Membuat array baru dengan semua elemen yang lulus uji yang diberikan oleh fungsi callback }

function _array_map (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] → _U) → List[_U]
{ Membuat array baru dengan hasil memanggil fungsi callback pada setiap elemen dalam array }

function _array_flat (array: List[Any], depth: Integer = 1, result: List[Any] = []) → List[Any]

{ Mengembalikan array baru dengan kedalaman yang ditentukan, menggabungkan sub-array hingga kedalaman yang ditentukan }

function _array_flat_map (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] → _U) → List[Any]
{ Mengembalikan array baru dengan setiap elemen diubah oleh fungsi callback dan kemudian diratakan satu tingkat }

function _array_join (array: List[Any], separator: String = ",") → String
{ Menggabungkan semua elemen array menjadi satu string dengan pemisah yang ditentukan }

function _array_reduce (array: List[_T], callback: Fungsi[_U, _T, Integer, List[_T]] → _U, initial: _U) → _U
{ Mengurangi array menjadi satu nilai dengan menjalankan fungsi callback pada setiap elemen, dari kiri ke kanan }

function _array_reduce_right (array: List[_T], callback: Fungsi[_U, _T, Integer, List[_T]] → _U, initial: _U) → _U
{ Mengurangi array menjadi satu nilai dengan menjalankan fungsi callback pada setiap elemen, dari kanan ke kiri }

procedure _array_reverse (array: List[_T])
{ Membalikkan elemen-elemen dalam array di tempat }
function _array_sort (array: List[_T], comparator: Fungsi[_T, _T] → Integer = lambda x, y: x - y) → List[_T]
{ Mengurutkan elemen-elemen dalam array menggunakan fungsi pembanding yang ditentukan }

function _array_to_reversed (array: List[_T]) → List[_T]
{ Mengembalikan array baru dengan elemen-elemen yang diurutkan dalam urutan terbalik }

function _array_to_sorted (array: List[_T], comparator: Fungsi[_T, _T] → Integer = lambda x, y: x - y) → List[_T]
{ Mengembalikan array baru dengan elemen-elemen yang diurutkan menggunakan fungsi pembanding yang ditentukan }

function _array_to_spliced (array: List[_T], start: Integer, deleteCount: Integer, *elements: _T) → List[_T]
{ Mengembalikan array baru dengan elemen-elemen yang telah dihapus dan/atau ditambahkan }

function _array_with (array: List[_T], index: Integer, value: _T) → List[_T]
{ Mengembalikan array baru dengan nilai baru pada indeks yang ditentukan }

{ FUNGSI/PROSEDUR PRIMORDIAL TUPLE }
function _tuple_with (tuple: _Tuple, **elements: Any) → _Tuple
{ Outputs a new tuple with the specified values replaced }

```
function _namedtuple_with (tuple: _NamedTuple, **elements: Any) → _NamedTuple
{ Outputs a new NamedTuple with the specified values replaced }
```

```
ALGORITMA PROGRAM UTAMA
_T ← TypeVar("_T")

_U ← TypeVar("_U")

_string_whitespace_characters ← ['\t', '\x0b', '\x0c', ' ', '\xa0', '\ufeff', '\u3000', '\u1680',
'\u2000', '\u2001', '\u2002', '\u2003', '\u2004', '\u2005', '\u2006', '\u2008', '\u2009', '\u200a',
'\u205f', '\u2007', '\u202f']

_string_line_terminators_characters ← ['\n', '\r', '\u2028', '\u2029']
```

```
{ REALISASI FUNGSI/PROSEDUR }
{ FUNGSI PRIMORDIAL STRING }
function _string_concat (strings : sequence of string) → string
 { menghasilkan hasil konkatenasi semua string dalam strings }
KAMUS LOKAL
  result : string
  string : string
ALGORITMA
  result ← ""
  for each string in strings do
      result ← result + string
  output result

function _string_slice (string : string, start : integer, end : integer = None) → string
{ menghasilkan substring dari string dimulai dari indeks start hingga sebelum indeks end }
KAMUS LOKAL
  result : string
  i : integer
ALGORITMA
  if end = None then
    end ← length(string)
  result ← ""
  i traversal [start..end-1]
    result ← result + string[i]
  output result

function _string_substring (string : string, start : integer, end : integer = None) → string
{ menghasilkan substring dari string dimulai dari indeks start hingga sebelum indeks end
dengan menggunakan _string_slice }
KAMUS LOKAL
ALGORITMA
  output _string_slice(string, start, end)

function _string_code_point_at (string : string, i : integer) → integer
{ menghasilkan kode titik untuk karakter pada indeks i dalam string }
KAMUS LOKAL
```

```
  codePoint : integer
ALGORITMA
  codePoint ← ord(string[i])
  output codePoint

function _string_index_of (string : string, search : string, position : integer = 0) → integer
{ menghasilkan indeks pertama dari kemunculan substring search dalam string mulai dari posisi tertentu }
KAMUS LOKAL
  stringLength : integer
  searchLength : integer
  i : integer
  j : integer
  valid : boolean
ALGORITMA
  stringLength ← length(string)
  searchLength ← length(search)
  if position < 0 then
    position ← 0
  if position > stringLength then
    position ← stringLength
  if searchLength = 0 then
    output position
  i traversal [position..stringLength – searchLength]
    valid ← True
    j traversal [0..searchLength – 1]
      if string[i + j] ≠ search[j] then
        valid ← False
        break
    if valid then
      output i
  output -1

function _string_last_index_of (string : string, search : string, position : integer = None) → integer
{ menghasilkan indeks terakhir dari kemunculan substring search dalam string mulai dari posisi tertentu }
KAMUS LOKAL
  stringLength : integer
  searchLength : integer
  i : integer
  j : integer
  valid : boolean

ALGORITMA
  stringLength ← length(string)
  searchLength ← length(search)
  if position = None then
  position ← stringLength
  if position < 0 then
```

```
  position ← 0
  if position > stringLength then
  position ← stringLength
  if searchLength = 0 then
  output position
  i traversal [position – searchLength..0]
    valid ← True
  j traversal [0..searchLength – 1]
  if string[i + j] ≠ search[j] then
  valid ← False
  break
  if valid then
  output i
  output -1

function _string_includes (string : string, search : string) → boolean
{ menghasilkan True jika search ditemukan dalam string, False jika tidak }
KAMUS LOKAL
ALGORITMA
  output _string_index_of(string, search) ≥ 0

function _string_starts_with (string : string, search : string) → boolean
{ menghasilkan True jika string dimulai dengan substring search, False jika tidak }
KAMUS LOKAL
  stringLength : integer
  searchLength : integer
  i : integer
ALGORITMA
  stringLength ← length(string)
  searchLength ← length(search)
  if stringLength < searchLength then
    output False
  i traversal [0..searchLength – 1]
    if string[i] ≠ search[i] then
      output False
  output True

function _string_ends_with (string : string, search : string) → boolean
{ menghasilkan True jika string diakhiri dengan substring search, False jika tidak }
KAMUS LOKAL
  stringLength : integer
  searchLength : integer
  i : integer
ALGORITMA
  stringLength ← length(string)
  searchLength ← length(search)
  if stringLength < searchLength then
    output False
  i traversal [0..searchLength – 1]
    if string[stringLength - searchLength + i] ≠ search[i] then
```

```
        output False
    output True

function _string_pad_start (string : string, length : integer, padding : string) → string
{ menambahkan padding di awal string sehingga panjangnya menjadi length }
KAMUS LOKAL
    additionalPadding : integer
ALGORITMA
    additionalPadding ← length - length(string)
    if additionalPadding ≤ 0 then
        output string
    output _string_slice(_string_repeat(padding, additionalPadding // length(padding)), 0,
additionalPadding) + string

function _string_pad_end (string : string, length : integer, padding : string) → string
{ menambahkan padding di akhir string sehingga panjangnya menjadi length }
KAMUS LOKAL
    additionalPadding : integer
ALGORITMA
    additionalPadding ← length - length(string)
    if additionalPadding ≤ 0 then
        output string
    output string + _string_slice(_string_repeat(padding, additionalPadding //
length(padding)), 0, additionalPadding)

function _string_repeat (string : string, count : integer) → string
{ menghasilkan string yang diulang sebanyak count kali }
KAMUS LOKAL
    result : string
    _ : integer
ALGORITMA
    result ← ""
    _ traversal [0..count – 1]
        result ← result + string
    output result

function _string_to_upper_case (string : string) → string
{ menghasilkan string dengan semua huruf diubah menjadi huruf besar }
KAMUS LOKAL
ALGORITMA
    output string.upper()

function _string_to_lower_case (string : string) → string
{ menghasilkan string dengan semua huruf diubah menjadi huruf kecil }
KAMUS LOKAL
ALGORITMA
    output string.lower()

function _string_trim_end (string: String) → String
{ Menghapus spasi dan pemisah baris dari akhir string }
```

```
KAMUS LOKAL
    trimIndex: Integer
    character: String
ALGORITMA
    trimIndex ← length(string) - 1
    while trimIndex ≥ 0 do
        character ← string[trimIndex]
        if not (_array_includes(_string_whitespace_characters, character) or
_array_includes(_string_line_terminators_characters, character)) then
            break
        trimIndex ← trimIndex - 1
    output _string_slice(string, 0, trimIndex + 1)

function _string_trim_start (string: String) → String
{ Menghapus spasi dan pemisah baris dari awal string }
KAMUS LOKAL
    stringLength: Integer
    trimIndex: Integer
    character: String
ALGORITMA
    stringLength ← length(string)
    trimIndex ← 0
    while trimIndex < stringLength do
        character ← string[trimIndex]
        if not (_array_includes(_string_whitespace_characters, character) or
_array_includes(_string_line_terminators_characters, character)) then
            break
        trimIndex ← trimIndex + 1
    output _string_slice(string, trimIndex)

function _string_trim (string: String) → String
{ Menghapus spasi dan pemisah baris dari kedua ujung string  }
KAMUS LOKAL
ALGORITMA
    output _string_trim_start(_string_trim_end(string))

function _string_replace (string: String, search: String, replacement: String/Function) →
String
{ Mengganti kemunculan pertama dari search dengan replacement dalam string  }
KAMUS LOKAL
    searchLength: Integer
    foundIndex: Integer
    foundString: String
    replacementString: String
ALGORITMA
    searchLength ← length(search)
    foundIndex ← _string_index_of(string, search)
    if foundIndex = -1 then
        output string
    foundString ← _string_slice(string, foundIndex, foundIndex + searchLength)
```

```
    if callable(replacement) then
        replacementString ← replacement(foundString, foundIndex, string, {})
    else
        replacementString ← replacement
    output _string_slice(string, 0, foundIndex) + replacementString + _string_slice(string,
foundIndex + searchLength)

function _string_replace_all (string: String, search: String, replacement: String/Function)
→ String
{ Mengganti semua kemunculan dari search dengan replacement dalam string }
KAMUS LOKAL
    stringLength: Integer
    searchLength: Integer
    result: String
    index: Integer
    foundIndex: Integer
    foundString: String
ALGORITMA
    stringLength ← length(string)
    searchLength ← length(search)
    result ← ""
    index ← 0
    while index < stringLength do
        foundIndex ← _string_index_of(string, search, index)
        if foundIndex = -1 then
            break
        foundString ← _string_slice(string, foundIndex, foundIndex + searchLength)
        result ← result + _string_slice(string, index, foundIndex)
        if callable(replacement) then
            result ← result + replacement(foundString, foundIndex, string, {})
        else
            result ← result + replacement
        index ← foundIndex + searchLength
    result ← result + _string_slice(string, index)
    output result

function _string_split (string: String, separator: String) → List<String>
{ Membagi string dengan separator dan mengembalikan daftar substring }
KAMUS LOKAL
    stringLength: Integer
    separatorLength: Integer
    result: List<String>
    index: Integer
    foundIndex: Integer
ALGORITMA
    stringLength ← length(string)
    separatorLength ← length(separator)
    result ← []
    index ← 0
    while index < stringLength do
```

```
        foundIndex ← _string_index_of(string, separator, index)
        if foundIndex = -1 then
            break
        _array_push(result, _string_slice(string, index, foundIndex))
        index ← foundIndex + separatorLength
    _array_push(result, _string_slice(string, index))
    output result


{ FUNGSI/PROSEDUR PRIMORDIAL ARRAY }
function _array_push (array: List, *elements) → Integer
{ Menambahkan satu atau lebih elemen ke akhir array dan mengembalikan panjang baru
array }
KAMUS LOKAL
    element: Any
ALGORITMA
    for each element in elements do
        append element to array
    output length(array)

function _array_pop (array: List) → Any
{ Menghapus elemen terakhir dari array dan mengembalikan elemen tersebut }
KAMUS LOKAL
ALGORITMA
    if length(array) = 0 then
        output None
    output remove last element from array

function _array_unshift (array: List, *elements) → Integer
{ Menambahkan satu atau lebih elemen ke awal array dan mengembalikan panjang baru
array }
KAMUS LOKAL
    element: Any
ALGORITMA
    _array_reverse(elements)
    for each element in elements do
        insert element at the start of array
    output length(array)

function _array_shift (array: List) → Any
{ Menghapus elemen pertama dari array dan mengembalikan elemen tersebut }
KAMUS LOKAL
ALGORITMA
    if length(array) = 0 then
        output None
    output remove first element from array

function _array_concat (*arrays: List) → List
{ Menggabungkan dua atau lebih array dan mengembalikan array baru }
KAMUS LOKAL
```

```
    array: List
ALGORITMA
    result ← []
    for each array in arrays do
        append all elements of array to result
    output result

function _array_slice (array: List, start: Integer = 0, end: Integer = None) → List
{ Mengembalikan salinan dangkal dari sebagian array menjadi objek array baru yang
dipilih dari start hingga end }
KAMUS LOKAL
ALGORITMA
    if end = None then
        end ← length(array)
    output copy of sub-array from start to end

function _array_splice (array: List, start: Integer, deleteCount: Integer = None, *elements)
→ List
{ Mengubah isi array dengan menghapus atau mengganti elemen yang ada dan/atau
menambahkan elemen baru di tempatnya }
KAMUS LOKAL
    deleted: List
    element: Any
ALGORITMA
    if deleteCount = None then
        deleteCount ← length(array) - start
    deleted ← []
    while deleteCount > 0 do
        if start ≥ length(array) then
            break
        append array[start] to deleted
        remove array[start]
        deleteCount ← deleteCount - 1
    _array_reverse(elements)
    for each element in elements do
        insert element into array at position start
    output deleted

function _array_copy_within (array: List, target: Integer, start: Integer, end: Integer =
None) → List
{ Menyalin sebagian array secara dangkal ke lokasi lain dalam array yang sama dan
mengembalikannya tanpa mengubah panjangnya }
KAMUS LOKAL
ALGORITMA
    if end = None then
        end ← length(array)
    i traversal [start .. end – 1]
        array[target + i - start] ← array[i]
    output array
```

```
function _array_every (array: List, callback: Function) → Boolean
{ Menguji apakah semua elemen dalam array lulus uji yang diimplementasikan oleh fungsi yang disediakan }
KAMUS LOKAL
   i: Integer
ALGORITMA
   i traversal [0 .. length(array) - 1 ]
      if callback(array[i], i, array) is not True then
         output False
   output True
function _array_some (array: List, callback: Function) → Boolean
{ Menguji apakah setidaknya satu elemen dalam array lulus uji yang diimplementasikan oleh fungsi yang disediakan }
KAMUS LOKAL
   i: Integer
ALGORITMA
   i traversal [0 .. length(array) – 1]
      if callback(array[i], i, array) is True then
         output True
   output False

function _array_index_of (array: List[_T], element: _T) → Integer
{ Mengembalikan indeks pertama di mana elemen ditemukan dalam array, atau -1 jika tidak ditemukan }
KAMUS LOKAL
   i: Integer
ALGORITMA
   i traversal [0 .. length(array) – 1]
      if array[i] ≠ element then
         continue
      output i
   output -1

function _array_last_index_of (array: List[_T], element: _T) → Integer
{ Mengembalikan indeks terakhir di mana elemen ditemukan dalam array, atau -1 jika tidak ditemukan }
KAMUS LOKAL
   i: Integer
ALGORITMA
   i traversal [length(array) - 1 .. 0]
      if array[i] ≠ element then
         continue
      output i
   output -1

function _array_includes (array: List[_T], element: _T) → Boolean
{ Memeriksa apakah array berisi elemen yang ditentukan }
KAMUS LOKAL
ALGORITMA
```

```
    output _array_index_of(array, element) ≥ 0

function _array_find (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] → Boolean)
→ _T
{ Mengembalikan elemen pertama dalam array yang lulus uji yang diberikan oleh fungsi
callback }
KAMUS LOKAL
    i: Integer
ALGORITMA
    i traversal [0 .. length(array) - 1]
        if not callback(array[i], i, array) then
            continue
        output array[i]
    output None

function _array_find_index (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] →
Boolean) → Integer
{ Mengembalikan indeks elemen pertama dalam array yang lulus uji yang diberikan oleh
fungsi callback }
KAMUS LOKAL
    i: Integer
ALGORITMA
    i traversal [0 to length(array) – 1]
        if not callback(array[i], i, array) then
            continue
        output i
    output -1

function _array_find_last (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] →
Boolean) → _T
{ Mengembalikan elemen terakhir dalam array yang lulus uji yang diberikan oleh fungsi
callback }
KAMUS LOKAL
    i: Integer
ALGORITMA
    i traversal [length(array) - 1 .. 0]
        if not callback(array[i], i, array) then
            continue
        output array[i]
    output None

function _array_find_last_index (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] →
Boolean) → Integer
{ Mengembalikan indeks elemen terakhir dalam array yang lulus uji yang diberikan oleh
fungsi callback }
KAMUS LOKAL
    i: Integer
ALGORITMA
    i traversal [length(array) - 1 .. 0]
        if not callback(array[i], i, array) then
```

```
            continue
        output i
    output -1

function _array_filter (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] → Boolean)
→ List[_T]
{ Membuat array baru dengan semua elemen yang lulus uji yang diberikan oleh fungsi
callback }
KAMUS LOKAL
    result: List[_T]
    i: Integer
ALGORITMA
    result ← []
    i traversal [0 .. length(array) – 1]
        if not callback(array[i], i, array) then
            continue
        append array[i] to result
    output result

function _array_map (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] → _U) →
List[_U]
{ Membuat array baru dengan hasil memanggil fungsi callback pada setiap elemen dalam
array }
KAMUS LOKAL
result: List[_U]
i: Integer
ALGORITMA
    result ← []
    i traversal [0 .. length(array) – 1]
        append callback(array[i], i, array) to result
    output result

function _array_flat (array: List[Any], depth: Integer = 1, result: List[Any] = []) →
List[Any]
{ Mengembalikan array baru dengan kedalaman yang ditentukan, menggabungkan sub-
array hingga kedalaman yang ditentukan }
KAMUS LOKAL
    i: Integer
ALGORITMA
    i traversal [0 .. length(array) – 1]
        if not hasattr(array[i], "__len__") then
            append array[i] to result
            continue
        _array_flat(array[i], depth - 1, result)
    output result
```

```
function _array_flat_map (array: List[_T], callback: Fungsi[_T, Integer, List[_T]] → _U)
→ List[Any]
{ Mengembalikan array baru dengan setiap elemen diubah oleh fungsi callback dan
kemudian diratakan satu tingkat }
KAMUS LOKAL
ALGORITMA
    output _array_flat(_array_map(array, callback))


function _array_join (array: List[Any], separator: String = ",") → String
{ Menggabungkan semua elemen array menjadi satu string dengan pemisah yang
ditentukan }
KAMUS LOKAL
    result: String
    arrayLength: Integer
    i: Integer
ALGORITMA
    result ← ""
    arrayLength ← length(array)
    i traversal [0 .. arrayLength – 1]
        result ← result + array[i]
        if i ≠ arrayLength - 1 then
            result ← result + separator
    output result


function _array_reduce (array: List[_T], callback: Fungsi[_U, _T, Integer, List[_T]] → _U,
initial: _U) → _U
{ Mengurangi array menjadi satu nilai dengan menjalankan fungsi callback pada setiap
elemen, dari kiri ke kanan }
KAMUS LOKAL
    accumulator: _U
    i: Integer
ALGORITMA
    accumulator ← initial
    i traversal [0 .. length(array) – 1]
        accumulator ← callback(accumulator, array[i], i, array)
    output accumulator

function _array_reduce_right (array: List[_T], callback: Fungsi[_U, _T, Integer, List[_T]]
→ _U, initial: _U) → _U
{ Mengurangi array menjadi satu nilai dengan menjalankan fungsi callback pada setiap
elemen, dari kanan ke kiri }
KAMUS LOKAL
    accumulator: _U
    i: Integer
ALGORITMA
    accumulator ← initial
    i traversal [length(array) – 1 .. 0]
        accumulator ← callback(accumulator, array[i], i, array)
    output accumulator
```

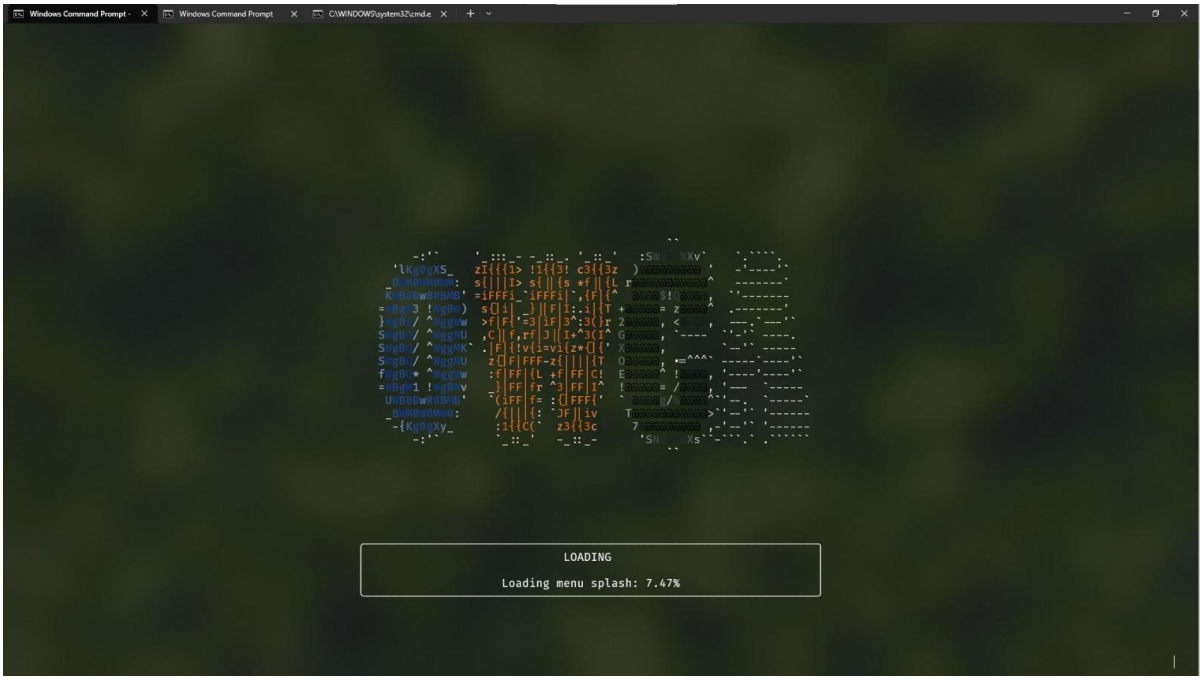
```
procedure _array_reverse (array: List[_T])
{ Membalikkan elemen-elemen dalam array di tempat }
KAMUS LOKAL
    arrayLength: Integer
    i: Integer
ALGORITMA
    arrayLength ← length(array)
    i traversal [0 .. arrayLength // 2]
        array[i] ← array[arrayLength - i - 1]


function _array_sort (array: List[_T], comparator: Fungsi[_T, _T] → Integer = lambda x,
y: x - y) → List[_T]
{ Mengurutkan elemen-elemen dalam array menggunakan fungsi pembanding yang
ditentukan }
KAMUS LOKAL
    arrayLength: Integer
    i: Integer
    temp: _T
    j: Integer
ALGORITMA
    arrayLength ← length(array)
    i traversal [1 to arrayLength – 1]
        temp ← array[i]
        j ← i
        while comparator(temp, array[j - 1]) < 0 do
            array[j] ← array[j - 1]
            j ← j - 1
        array[j] ← temp
    output array

function _array_to_reversed (array: List[_T]) → List[_T]
{ Mengembalikan array baru dengan elemen-elemen yang diurutkan dalam urutan
terbalik }
KAMUS LOKAL
ALGORITMA
    array ← _array_slice(array)
    _array_reverse(array)
    output array


function _array_to_sorted (array: List[_T], comparator: Fungsi[_T, _T] → Integer =
lambda x, y: x - y) → List[_T]
{ Mengembalikan array baru dengan elemen-elemen yang diurutkan menggunakan fungsi
pembanding yang ditentukan }
KAMUS LOKAL
ALGORITMA
    array ← _array_slice(array)
    _array_sort(array, comparator)
    output array
```

```
function _array_to_spliced (array: List[_T], start: Integer, deleteCount: Integer, *elements:
_T) → List[_T]
{ Mengembalikan array baru dengan elemen-elemen yang telah dihapus dan/atau
ditambahkan }
KAMUS LOKAL
ALGORITMA
   array ← _array_slice(array)
   _array_splice(array, start, deleteCount, *elements)
   output array

function _array_with (array: List[_T], index: Integer, value: _T) → List[_T]
{ Mengembalikan array baru dengan nilai baru pada indeks yang ditentukan }
KAMUS LOKAL
ALGORITMA
   array ← _array_slice(array)
   array[index] ← value
   output array

{ FUNGSI/PROSEDUR PRIMORDIAL TUPLE }
function _tuple_with (tuple: _Tuple, **elements: Any) → _Tuple
{ Outputs a new tuple with the specified values replaced }
KAMUS LOKAL
   elements: list[Any]
   key: String
   value: Any
ALGORITMA
   elements ← []
   i traversal [0 .. length(tuple) – 1]
      elements[i] ← tuple[i]
   for key, value in elements do
      if not _string_starts_with(key, "_") then
         continue
      elements[int(_string_slice(key, 1))] ← value
   output (*elements,)

function _namedtuple_with (tuple: _NamedTuple, **elements: Any) → _NamedTuple
{ Outputs a new NamedTuple with the specified values replaced }
KAMUS LOKAL
   tupleType: Type[NamedTuple]
   key: String
   value: Any
ALGORITMA
   tupleType ← type(tuple)
   elements ← dict()
   for key in tupleType._fields do
      elements[key] ← getattr(tuple, key)
   for key, value in elements do
      elements[key] ← value
```

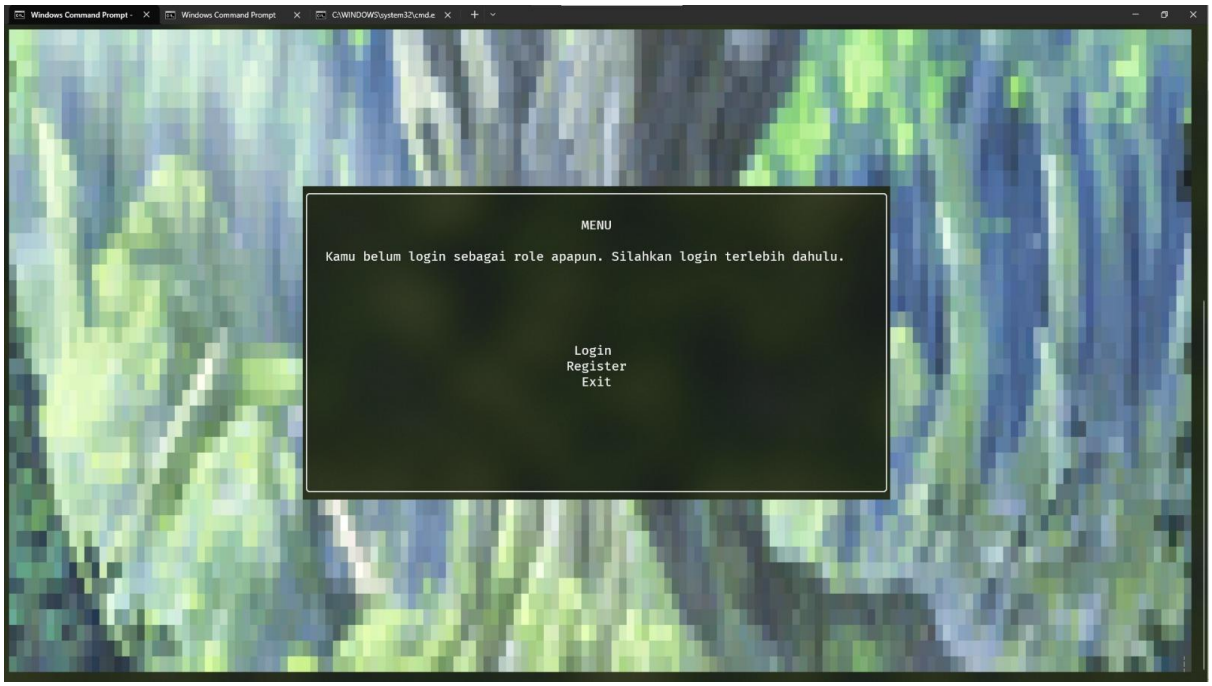
```
output tupleType(**elements)
```

VII. HASIL PENGUJIAN PROGRAM BERDASARKAN FITUR

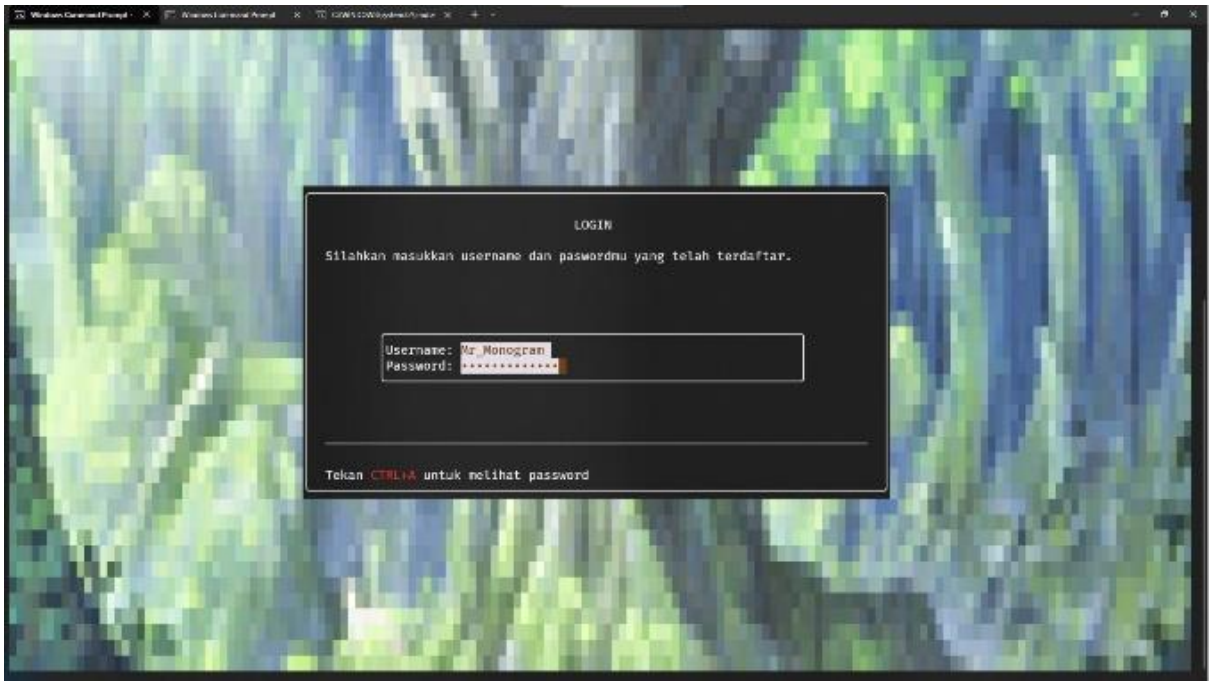
Gambar VII.I Menu Starting



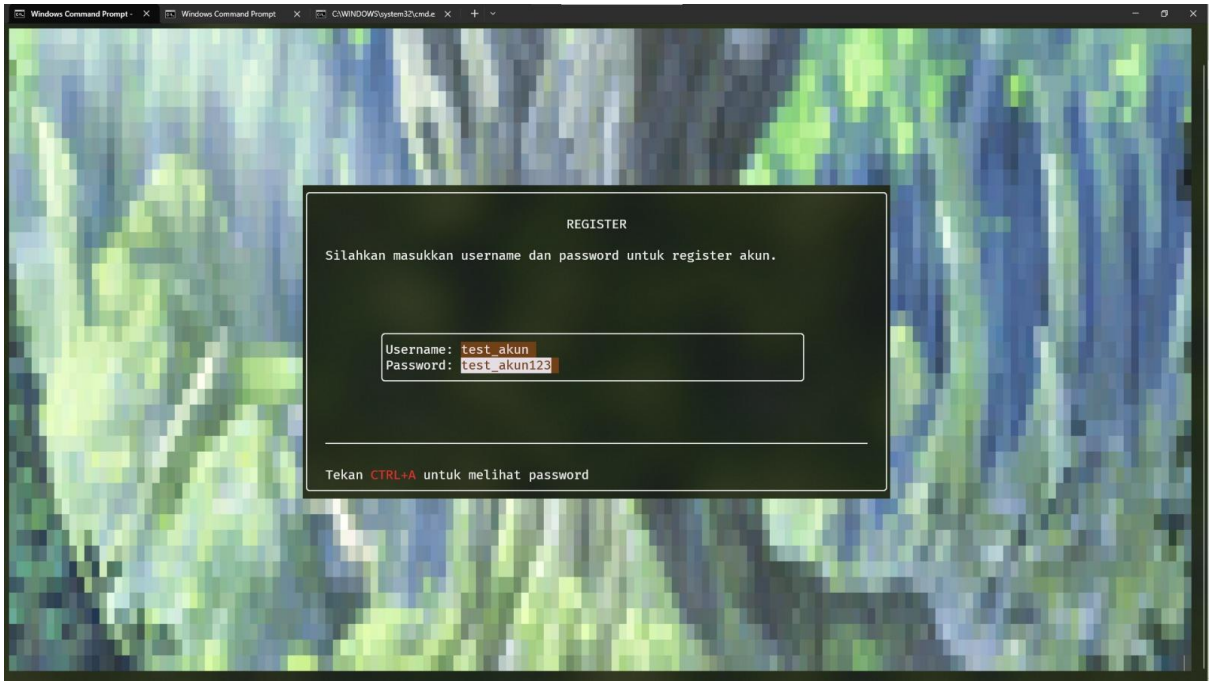
Gambar VII.II Menu Help Sebelum Login



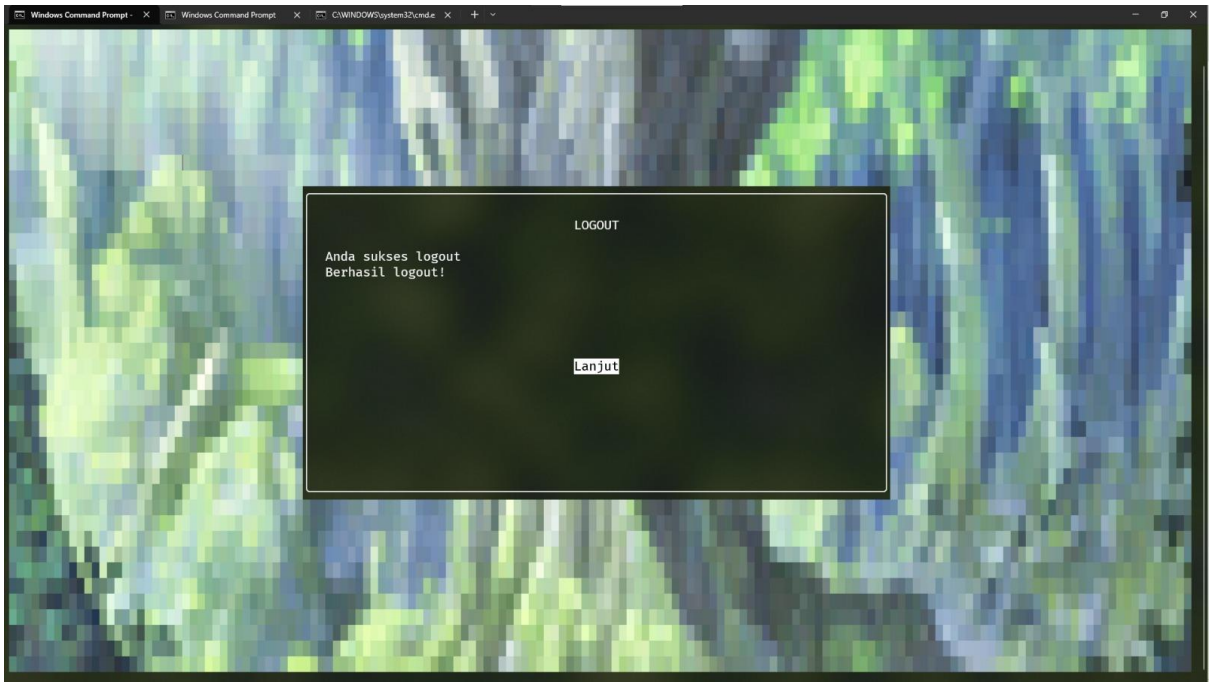
Gambar VII.III Input Login



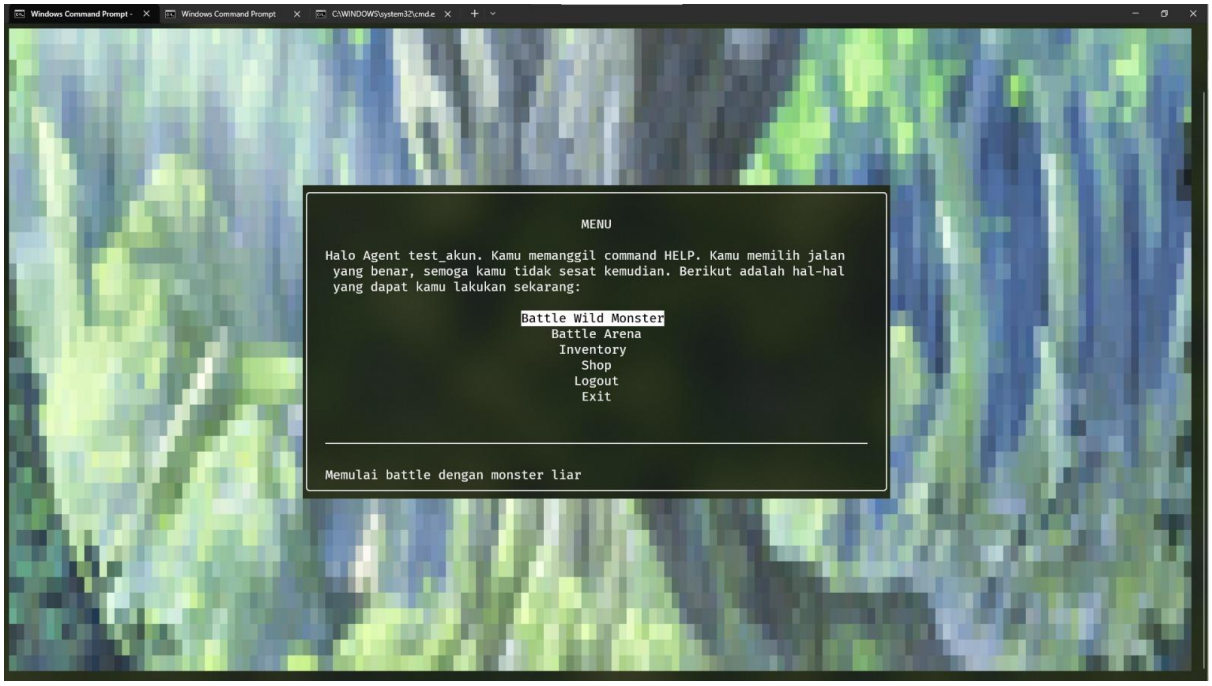
Gambar VII.IV Input Register



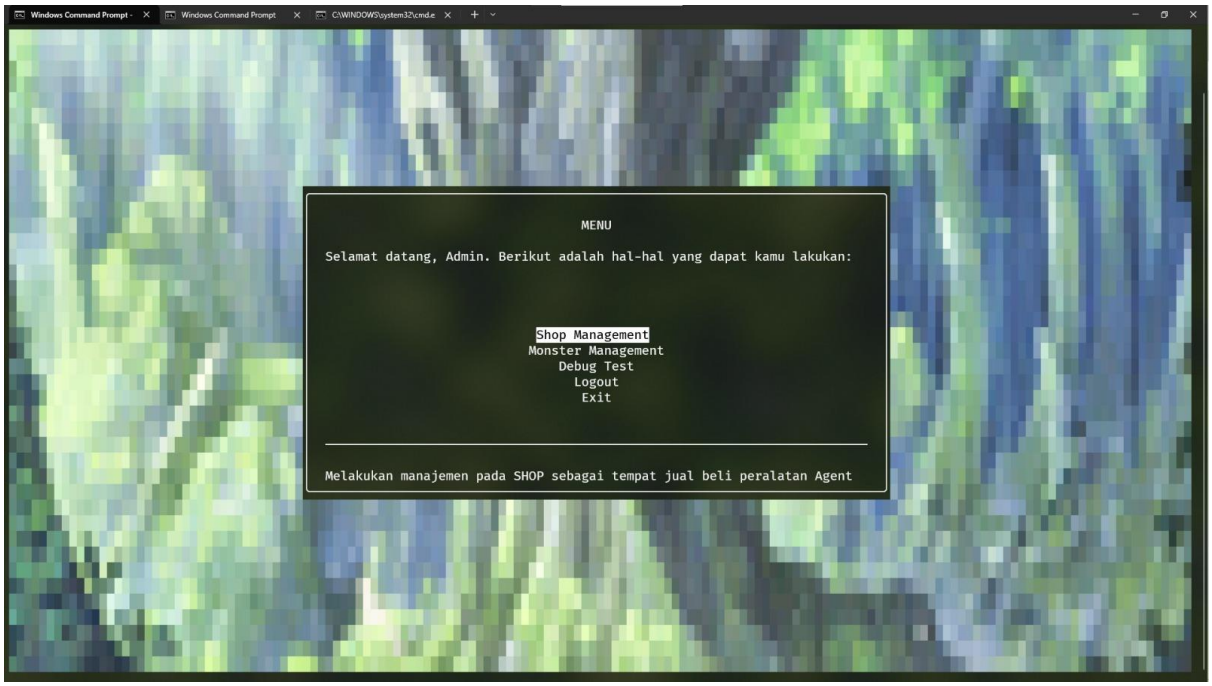
Gambar VII.V Logout



Gambar VII.VI Menu Help Sebagai Agent

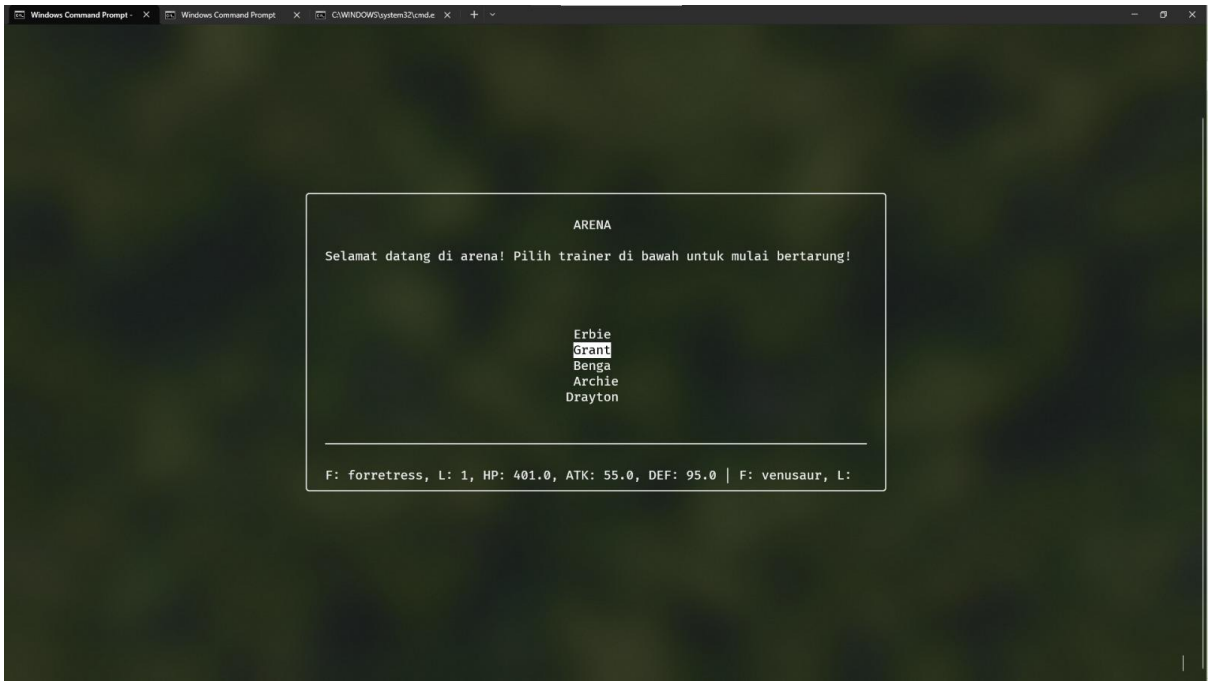


Gambar VII.VII Menu Help Sebagai Admin



Gambar VII.VIII Tampilan Battle

Gambar VII.IX Tampilan Arena



VIII. LAMPIRAN

Lampiran 1. Form Asistensi

Nomor Asistensi : **1**

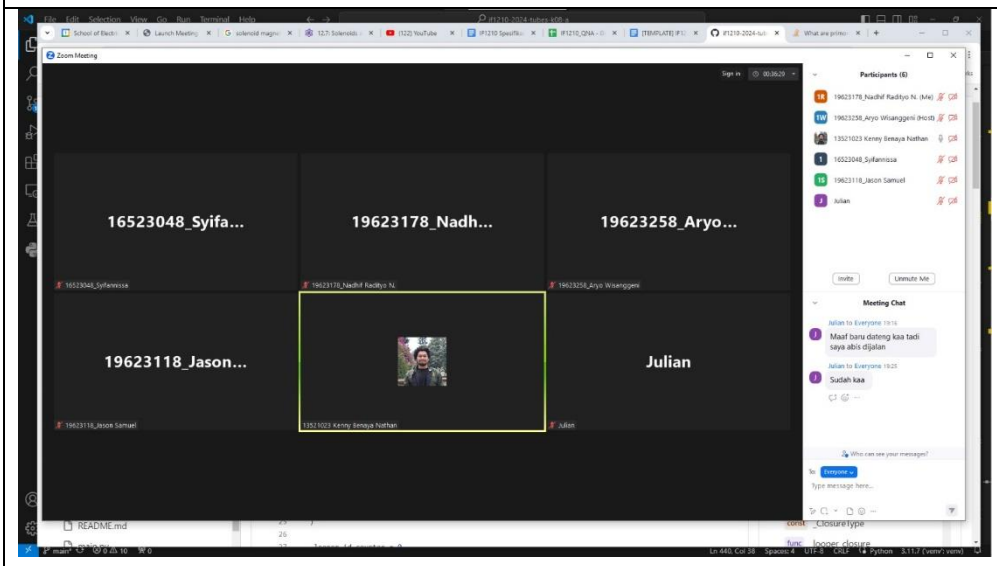
No. Kelompok/Kelas : **K08-A**

Tanggal asistensi : **1 May 2024**

Anggota kelompok		**NIM / Nama (Hanya yang Hadir)**
	1	**19623258/ Aryo Wisanggeni**
	2	**16523048/ Syifanissa Nafisalia Kuncoro**
	3	**16523178/ Julian Benedict**
	4	**19623178/ Nadhif Radityo Nugroho**
	5	**19623118/ Jason Samuel**
	6	
Asisten pembimbing		**NIM / Nama**
		13521023/ Kenny Benaya Nathan

Catatan Asistensi:

Rangkuman Diskusi
Ada bagian yang diperbaiki yaitu: **commit c5223: fix(utils): Convert string typing to OG typing** **commit d7a17: fix(utils): Fix primordials import** **commit 20ae4: feat: Modify csv parser to use primordials** **commit e8556: feat(utils): Add utils as editable pip package** **commit ed592: test: add csv files for testing code** **commit 20f0f: feat: add feature to load data (F14)** **commit c3d39: feat(utils): Fixup coroutines' typing and basic main loop** **commit ac08b: feat(utils): Cleanup primordials and implemented string_split** **commit cabec: feat(utils): Added basic custom coroutines mechanism** **commit 4c8c4: fix(utils): Prevent generic type definition variables are not escaping module encapsulation** **commit a073f: feat(utils): Move primordials into its own directory** **commit 36c1b: feat(utils): Implemented array primordials and rearrange utils folder** **commit 92c3b: feat(utils): Initial string basic operations** **commit 272bf: Added RNG module using LCG** **commit 2168c: small fix for lcg commit** **commit d41b7: Added module for RNG using LCG**

Dan pada diskusi kali ini belum terlalu banyak yang dibahas dikarenakan jarak waktu antara mulai tugas besar ini dengan jadwal asistensi terlalu dekat. Sehingga, masih belum membaca banyak spesifikasi dari modul. Namun, gambaran bagaimana tugas ini sudah ada dan apabila terdapat kebingungan untuk side cases dalam spesifikasi dapat ditanyakan ke kakaknya atau langsung ke QnA.
Tindak Lanjut
Disarankan oleh ka Kenny bahwa kita menggunakan tubes ini sebagai kesempatan untuk belajar. Kak Kenny menjelaskan bagaimana pentingnya tugas besar sebagai sarana untuk belajar dan agar bisa saling bermanfaat untuk semuanya. Dengan ini, tugas tidak terasa menjadi terlalu sulit dan bisa share pengalaman belajar bersama. **Selanjutnya Kak Kenny juga menyarankan kami untuk membagi tugas, supaya rata dan efektif. Pada akhir meeting, kami menentukan mekanisme pembagian tugas. Mekanisme pembagian tugas dengan cara membuat file inisial yang mana nanti tiap-tiap anggota langsung mengedit file tersebut. Hal tersebut agar mencegah merge conflict dan membuat akses management data yang efisien dan teratur.** **Dilanjutkan dengan pembahasan spesifikasi program utama bagian spesifikasi F05, F07, F14. Dilanjutkan dengan pembahasan spesifikasi bonus, yang mana kami menentukan untuk mengerjakan bonus B01, B02, BX1 (Rencana critical damage), BX2 (Rencana ascii art owca).**
Dokumentasi


Lampiran 2. Form Asistensi 2

Nomor Asistensi : **2**

No. Kelompok/Kelas : **K08-A**

Tanggal asistensi : **11 May 2024**

Anggota kelompok		**NIM / Nama (Hanya yang Hadir)**
	1	**19623258/ Aryo Wisanggeni**
	2	**16523048/ Syifanissa Nafisalia Kuncoro**
	3	**16523178/ Julian Benedict**
	4	**19623178/ Nadhif Radityo Nugroho**
	5	**19623118/ Jason Samuel**
	6	
Asisten pembimbing		**NIM / Nama**
		13521023/ Kenny Benaya Nathan

Catatan Asistensi:

Rangkuman Diskusi
Pada asistensi pada kali ini ka Kenny selaku asisten pembimbing kami melakukan check up kepada setiap anggota kelompok, yaitu di check tiap anggota melakukan bagian apa, dan pembagiannya sebagai berikut: **F00 - Random Number Generator - Aryo** ✅ **F01 - Register - Aryo** ✅ **F02 - Login - Aryo** ✅ **F03 - Logout - Aryo** ✅ **F04 - Menu & Help - Julian** ✅ **F05 - Monster - syifa** **F06 - Potion - Jason** **F07 - Inventory - Jason** **F08 - Battle - Nadhif** ✅ **(belum commit)** **F09 - Arena - Nadhif** ✅ **F10 - Shop & Currency - Julian** ✅

F11 - Laboratory - Julian ✅

F12 - Shop Management - syifa ✅

F13 - Monster Management - syifa ✅

F14 - Load - Nadhif ✅

F15 - Save - Nadhif ✅

F16 - Exit - Jason ✅

B01 - Git Best Practice ✅

B02 - Typing ✅

B03 - Monster Ball

B04 - JACKPOT!

B05 - Peta Kota Danville

BX1 - Critical Damage

BX2 - Ascii Art - Aryo ✅

BX3 - Console UI – Nadhi

Tindak Lanjut

Setiap daripada kami memiliki kesulitannya masing-masing sehingga kami berkonsultasi dengan Kak Kenny. Dan Kak Kenny juga menyampaikan beberapa pendapat dirinya mengenai solusi apa yang harus kita lakukan.

Dokumentasi