

# Spesifikasi Tugas Besar

## IF1210 Algoritma & Pemrograman 1

### 2025

Tim Asisten Lab Pemrograman 2022

Versi	: <b>1</b>
Tgl. Revisi Terakhir	: 10 Mei 2025
Deadline	: 11 Mei 2025, 12.10 WIB (Milestone 1) 31 Mei 2025, 12.10 WIB (Milestone 2)

## Daftar Revisi

1. April 28, 9.56 ~ Perubahan Penomoran Spesifikasi Program Utama
2. April 28, 20.23 ~ Penambahan ketentuan pengumpulan milestone
3. April 30, 21.38 ~ Penambahan **F10** sebagai spesifikasi wajib dan perubahan fungsi **F18** - Exit untuk STI
4. Mei 1, 21:51 - Penambahan dan klarifikasi informasi terkait jaminan bahwa file eksternal pasti terdefinisi
5. Mei 8, 19.29 - Penambahan ketentuan pengumpulan laporan pada tiap milestone.
6. Mei 10, 17.35 - Penambahan ketentuan pada **F08** terkait algoritma searching

# Daftar Isi

<b>Daftar Revisi</b>	<b>2</b>
<b>Daftar Isi</b>	<b>3</b>
<b>Deskripsi Persoalan</b>	<b>5</b>
<b>Spesifikasi Program</b>	<b>8</b>
<b>Spesifikasi Program Utama</b>	<b>9</b>
F00 – Rencana Implementasi	9
F01 – Login	9
F02 – Register Pasien	10
F03 – Logout	11
F04 – Lupa Password	11
F05 – Menu & Help	12
F06 – Denah Rumah Sakit	14
F07 – Lihat User	14
F08 – Cari User	15
F09 – Lihat Antrian	17
F10 – Tambah Dokter	17
F11 – Diagnosis	18
F12 – Ngobatin	19
F13 – Aku boleh pulang ga, dok 🥺 ?	20
F14 – Daftar Check-Up	21
F15 – Antrian Saya!	23
F16 – Minum Obat	23
F17 – Minum Penawar	24
F18 – Exit	25
<b>Spesifikasi Program Tambahan (STI)</b>	<b>26</b>
S01 – Denah Rumah Sakit	26
S02 – Lihat Antrian	27
<b>Spesifikasi Program Tambahan (IF &amp; EL &amp; EB)</b>	<b>28</b>
D01 – Denah Rumah Sakit	28
D02 – Lihat Antrian	29
D03 – Load	30
D04 – Save	31
<b>Spesifikasi Bonus</b>	<b>33</b>

B01 – Git Best Practice	33
B02 – Denah Dinamis	35
B03 – Aura!	36
B04 – Banarich!!!!!! 🍌	38
B05 – Dead or Alive?!	40
B06 – Mainin Antrian	41
BXX – Kreativitas	43
<b>Struktur Data File</b>	<b>44</b>
File User (user.csv)	46
File Penyakit (penyakit.csv)	47
File Obat (obat.csv)	48
File Obat – Penyakit (obat_penyakit.csv)	49
<b>Ketentuan dan Batasan</b>	<b>50</b>
A. Ketentuan Umum	50
B. Batasan Pengerjaan	50
C. Jadwal Pelaksanaan Kegiatan	51
D. Pengerjaan Tugas Besar	51
<b>Pengumpulan dan Deliverables</b>	<b>53</b>
<b>Panduan Instalasi WSL + Makefile</b>	<b>55</b>
A. Windows 10/11	55
B. Mac	56
<b>Referensi</b>	<b>57</b>
<b>Extras</b>	<b>58</b>

## Deskripsi Persoalan



*"Kehidupan baru Gro, sang penjahat yang telah bertobat."*

---

Kehidupan di kompleks bawah tanah rahasia milik Gro telah memasuki babak baru yang penuh gejolak, meski bukan lagi gejolak dari rencana menguasai dunia. Sejak Gro bergabung dengan *Anti-Villain League* (AVL) bersama istrinya, Luyi, fokus utama telah bergeser dari kejahatan super menjadi... yah, mencoba menjaga agar rumah tangga tidak runtuh di tengah kekacauan yang diciptakan oleh ratusan Nimons 😊. Makhluk-makhluk kuning kecil berbentuk kapsul dengan loyalitas yang tak tergoyahkan dan kecenderungan alami untuk menimbulkan bencana adalah variabel X dalam kehidupan Gro. Mereka adalah pasukan pekerja, teman bermain bagi ketiga putrinya, dan sumber sakit kepala utama bagi siapa saja yang mencoba menjaga keteraturan. Laboratorium canggih yang dulu melahirkan sinar penyusut dan roket antar planet kini lebih sering menjadi tempat eksperimen selai atau arena pertempuran bantal Nimon skala besar.

Di tengah perubahan ini, Dr. Neroifa, ilmuwan brilian di balik sebagian besar teknologi (dan masalah) Gro, mulai merasa sedikit... terpinggirkan. Usianya mungkin bertambah, pendengarannya semakin selektif, tetapi otaknya masih menginginkan tantangan besar. Memperbaiki pemanggang roti atau merancang sistem keamanan rumah yang lebih baik (yang selalu berhasil ditembus oleh para Nimon) tidak memberikan kepuasan yang sama seperti menciptakan mesin kiamat mini. Ia merindukan sebuah proyek besar, sesuatu yang akan menguji batas kemampuannya. Pengamatan sehari-harinya terhadap para Nimon memberinya banyak bahan pemikiran. Ia melihat Kebin mencoba mengatur regu pembersih dengan hasil yang beragam, Stewart yang terus-menerus kehilangan gitar ukulelanya (seringkali karena digunakan Nimon lain sebagai dayung), dan Pop yang tak henti-hentinya memeluk segala sesuatu yang bergerak maupun tidak.

Namun, yang paling menarik perhatian Dr. Neroifa adalah frekuensi insiden medis yang tampaknya sepele namun terus-menerus terjadi. Hampir setiap hari ada saja Nimon yang terpeleset genangan jus pisang, tersengat listrik ringan saat mencoba 'membantu' memperbaiki kabel, atau menelan benda-benda yang sama sekali bukan makanan – kancing, koin, komponen robot kecil, bahkan sesekali kunci pas mini milik Dr. Neroifa sendiri. Penanganan insiden ini selalu serampangan. Gro mungkin akan memberikan plester, Luiy mencoba menenangkan dengan lagu, sementara Dr. Neroifa, jika sempat, akan melakukan intervensi cepat dengan alat-alat laboratoriumnya yang kurang cocok untuk prosedur medis. Tidak ada catatan, tidak ada tindak lanjut yang konsisten, hanya serangkaian solusi darurat. Ia menyadari pola yang mengkhawatirkan: para Nimon ini, meskipun tangguh, sering kali berada dalam bahaya karena kecerobohan mereka sendiri dan kurangnya sistem perawatan kesehatan yang layak.

"Ini tidak bisa diterima!" gumam Dr. Neroifa suatu malam, setelah menyaksikan tiga Nimon memerlukan bantuan karena mencoba menggunakan penyedot debu bertenaga roket sebagai wahana hiburan. "Mereka membutuhkan perawatan yang sistematis! Fasilitas yang tepat! Sebuah..." Ide itu menghantamnya seperti sengatan listrik (sesuatu yang sering dialami para Nimon). "Sebuah Rumah Sakit!" Bukan sekadar pos P3K, tapi sebuah pusat medis yang lengkap, dirancang khusus untuk menangani segala macam penyakit dan cedera unik yang hanya bisa dialami oleh Nimon. Visinya berkembang pesat: ruang-ruang pemeriksaan yang ramah Nimon, peralatan diagnostik canggih yang bisa membedakan antara sakit perut biasa dan kasus menelan kunci inggris, serta apotek yang menyediakan obat-obatan khusus (seperti 'Sirup Penenang Rasa Pisang' atau 'Tablet Penetral Efek Ramuan Anti-Gravitasi').

Dengan antusiasme yang baru ditemukan, Dr. Neroifa mulai merancang dan bahkan mengawasi pembangunan awal fasilitas tersebut di bagian laboratorium yang luas dan tidak terpakai, dengan bantuan para Nimon yang bekerja keras (meski seringkali salah arah). Namun, seiring bentuk fisik rumah sakit mulai terwujud, Dr. Neroifa dihadapkan pada masalah yang jauh lebih kompleks daripada sekadar membangun dinding atau memasang mesin sinar-X mini. Masalahnya adalah bagaimana mengelola semuanya. Bagaimana cara mendaftarkan ratusan pasien Nimon yang mungkin datang bersamaan setelah insiden ledakan soda eksperimental? Bagaimana melacak dokter Nimon mana yang sedang bertugas, spesialisasi mereka, dan pasien mana yang sedang mereka tangani? Bagaimana mengelola antrian panjang pasien yang menunggu pemeriksaan atau pengobatan?

Kekacauan mulai terlihat jelas. Upaya awal untuk membuat sistem manual gagal total. Papan tulis Kebin selalu berakhir penuh coretan atau bekas gigitan, nomor antrean pisang Stewart selalu hilang, dan meja informasi yang dijaga Pop hanya menawarkan pelukan, bukan solusi administratif. Stok obat-obatan unik mulai tercampur aduk, sulit untuk mengetahui apa yang tersedia, apa yang hampir habis, atau apa yang sudah kedaluwarsa. Data pasien, jika ada, ditulis di catatan-catatan acak yang mudah hilang atau tidak terbaca. Dr. Neroifa menyadari bahwa kejeniusannya dalam menciptakan perangkat keras tidak membantunya dalam merancang sistem organisasi informasi yang dibutuhkan. Ia bisa membangun robot bedah, tapi ia tidak tahu cara membuat sistem yang bisa menjadwalkan operasi, melacak riwayat pasien, atau mengelola inventaris secara efisien. Rumah sakit impiannya berisiko menjadi pusat kekacauan medis terbesar di dunia sebelum sempat merawat satu pasien pun.

Frustrasi mencapai puncaknya. Dr. Neroifa tahu ia membutuhkan bantuan. Ia memerlukan otak-otak cerdas yang memahami cara kerja sistem, cara mengelola aliran informasi yang kompleks, dan cara membawa keteraturan pada kekacauan. Ia membutuhkan seseorang yang bisa merancang dan membangun jantung digital rumah sakit ini – sebuah sistem manajemen yang komprehensif. Dan entah bagaimana, melalui saluran yang hanya diketahui olehnya, ia mengetahui tentang kalian, para mahasiswa IF1210. Kalian, dengan pemahaman kalian tentang logika dan struktur pemecahan masalah, adalah harapan terakhir Dr. Neroifa. Tugas kalian adalah mengubah kekacauan ini menjadi sistem yang terorganisir, memastikan setiap Nimons mendapatkan perawatan yang mereka butuhkan, dan membantu Dr. Neroifa mewujudkan visinya akan fasilitas medis yang (relatif) teratur. Nasib kesehatan para Nimons ada di tangan kalian!

# Spesifikasi Program

Terdapat kebutuhan fungsional wajib yang diperlukan oleh kalian sebagai Nimons, seperti yang tertera di bawah. Tampilan atau interface dari sistem dibebaskan, silahkan berkreasi; output tidak harus persis seperti contoh, yang penting spesifikasi terpenuhi. Kreativitas interface akan menjadi pertimbangan penilaian. Perlu diperhatikan bahwa yang menjadi penekanan bukan hanya pada **alur jalannya program**, tetapi juga pada **validasi aksi** yang dilakukan.

Banyaknya fungsionalitas yang wajib diimplementasikan bukan berarti program yang akan dibuat juga akan panjang dan kompleks. Dengan pembuatan program yang **modular** dengan **fungsi/prosedur yang jelas**, program dapat dibuat secara relatif lebih **singkat dan sederhana**. Silakan pahami spesifikasi Tugas Besar ini dengan baik sebelum melakukan pengerjaan. Ingat, ini merupakan Tugas Besar berkelompok, sehingga aturlah sedemikian mungkin agar tugas ini tidak dibebankan kepada beberapa orang saja.

## Notes:

Untuk mempermudah pengerjaan tugas besar, kami telah memberikan contoh pada masing-masing spesifikasi fungsional. Namun, keluaran dan masukan program dapat disesuaikan dengan kreativitas masing-masing. Adapun beberapa informasi tambahan terkait contoh yang kami berikan adalah sebagai berikut:

- Teks Normal berarti output
- **Teks Bold** berarti input
- **Teks Biru** berarti variabel tertentu

**WARNING: CICIL Pengerjaan dari jauh-jauh hari dan kerjakan dengan konsisten. Jika mengerjakan dekat dengan deadline, sangat beresiko tugas besar ini tidak selesai dengan baik!**



# Spesifikasi Program Utama

## F00 – Rencana Implementasi

Silakan buat sebuah rencana implementasi ADT dan rencana implementasi materi dalam dokumen. Rencana implementasi ini nantinya akan kalian jadikan panduan pengerjaan program. Berikut beberapa materi atau ADT yang harus kalian implementasikan:

- ADT Sederhana
- ADT List
- ADT Linked List (List dengan struktur berkait)
- ADT Matrix (Khusus IF & EL & EB)
- ADT Set
- ADT Map
- ADT Stack
- ADT Queue
- File External (Khusus IF & EL & EB)
- Fungsi & Prosedur
- Array Search, Sort, Filter (Khusus STI -> wajib ada binary search)

Notes:

Kalian diperbolehkan menggabungkan dua atau lebih ADT & materi dalam 1 implementasi

## F01 – Login

### Akses: belum Login

Sebelum memasuki sistem, pengguna harus mendaftarkan akunnya terlebih dahulu. Seorang pengguna baru harus memasukkan *username* dan *password* yang akan digunakan. Login dapat dilakukan oleh manager, dokter, dan pasien. Tetapi register hanya bisa dilakukan untuk pasien (dijelaskan lagi di F02). Manager hanya ada 1 dan sudah ada dari awal program dimulai, sedangkan pasien dan dokter awalnya 0 tetapi akan bisa bertambah banyak setelah registrasi (Pasien) atau ditambahkan (Pasien).

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja

# Kasus 1: Login sebagai Manager
>>> LOGIN
Username: nimonsslatte
```

Password: <b>nimonatutgajah23</b> Selamat pagi Manager <b>nimonsslatte!</b>
# Kasus 2: Login sebagai Dokter >>> <b>LOGIN</b> Username: <b>Neroifa</b> Password: <b>Neroifa123</b> Selamat pagi Dokter <b>Neroifa!</b>
# Kasus 3: Login sebagai Pasien >>> <b>LOGIN</b> Username: <b>GRO</b> Password: <b>NeroifaCantik</b> Selamat pagi <b>GRO!</b> Ada keluhan apa ?
# Kasus 4: Tidak ada username yang terdaftar >>> <b>LOGIN</b> Masukkan username: <b>NONEXISTENT_USER</b> Masukkan password: <b>passwordrandom</b> Tidak ada Manager, Dokter, atau pun Pasien yang bernama <b>NONEXISTENT_USER!</b>
# Kasus 5: Kasus password salah >>> <b>LOGIN</b> Masukkan username: <b>nimonsganteng</b> Masukkan password: <b>wrongpassword</b> Username atau password salah untuk pengguna yang bernama <b>nimonsganteng!</b>

## F02 – Register Pasien

### Akses: belum login

Pengguna yang belum login dapat melakukan registrasi menjadi seorang pasien. Perlu menjadi catatan di sini bahwa registrasi hanya bisa dilakukan untuk Pasien. Manager sudah ada dari awal dan jika ingin menambah dokter, bisa dilihat pada [F08](#). Khusus untuk fitur ini, silakan buat proses validasi Username unik dengan menggunakan struktur data **Set**. Validasi keunikan username dilakukan secara **case-insensitive**.

Notes:

- Username hanya tersusun dari kombinasi huruf besar dan huruf kecil.

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
# Kasus 1: Pasien bernama Denis belum ada
```

```
>>> REGISTER
```

```
Username: Denis
```

```
Password: aditgimananihdi
```

```
Pasien Denis berhasil ditambahkan!
```

```
>>> LOGIN
```

```
Username: Denis
```

```
Password: aditgimananihdi
```

```
Selamat pagi Denis! Ada keluhan apa ?
```

```
# Kasus 2: Pasien bernama Denis sudah ada.
```

```
>>> REGISTER
```

```
Username: Denis
```

```
Password: aditgimananihdi
```

```
Registrasi gagal! Pasien dengan nama Denis sudah terdaftar.
```

## F03 – Logout

### Akses: Manager, Dokter, Pasien

Setiap pengguna yang sedang *login* dapat melakukan prosedur ini untuk keluar dari akun yang sedang dipakai. Setelah melakukan *logout*, pengguna kehilangan akses dari akun sebelumnya dan dapat melakukan *login* lagi menggunakan akun yang berbeda. Prosedur ini hanya dapat dipanggil ketika sedang ada akun yang *logged in*.

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
```

```
# Kasus 1: sedang dalam keadaan logged in
```

```
>>> LOGOUT
```

```
# Keluar dari akun
```

```
Sampai jumpa!
```

```
# Kasus 2: sedang dalam keadaan belum logged in
```

```
>>> LOGOUT
```

```
Logout gagal!
```

```
Anda belum login, silahkan login terlebih dahulu sebelum melakukan logout
```

## F04 – Lupa Password

### Akses: Manager, Dokter, Pasien

User (Manager, Dokter, Pasien) dapat melakukan update password apabila lupa password yang didaftarkan pada akun tersebut. Untuk mempermudah, validasi user dilakukan dengan

melakukan pengecekan kode unik dari user. Kode unik merupakan **Run-Length Encoding** dari username user.

Contoh implementasi **Run-Length Encoding**:

**"AABBCCCDEEEFFGHIJJK" → "2A2B3CD3EFGHIJ2K"**

```
# Kasus 1: Manager, Dokter, atau Pasien bernama Jeffreey ada
```

```
>>> LUPA_PASSWORD
```

```
Username: Jeffreey
```

```
Kode Unik: Je2fr2ey
```

```
Halo Dokter Jeffreey, silakan daftarkan ulang password anda!
```

```
Password Baru: JR1234
```

```
>>> LOGIN
```

```
Username: Jeffreey
```

```
Password: JR1234
```

```
Selamat pagi Dokter Jeffreey!
```

```
# Kasus 2: Manager, Dokter, atau Pasien bernama NONEXISTENT_USER tidak ada
```

```
>>> LUPA_PASSWORD
```

```
Username: NONEXISTENT_USER
```

```
Kode Unik: NONEXISTENT_USER
```

```
Username tidak terdaftar!
```

```
# Kasus 3: Kode unik untuk username nimonsslatte bukan nimonsslatte tetapi adalah nimon2sla2te
```

```
>>> LUPA_PASSWORD
```

```
Username: nimonsslatte
```

```
Kode Unik: nimonsslatte
```

```
Kode unik salah!
```

## F05 – Menu & Help

### Akses: Manager, Dokter, Pasien, Belum Login

Dokter, Pasien, dan Manager terkadang bingung dengan apa yang bisa mereka lakukan di rumah sakit tersebut. Akhirnya, Nimons memiliki sebuah ide cemerlang untuk mensosialisasikan sebuah *command* yang dapat membantu menjadi petunjuk mereka, yaitu **HELP**. *Command HELP* disosialisasikan oleh Nimons dengan memasang banner di seluruh bagian rumah sakit, sehingga seluruh orang di rumah sakit dapat melihat *command HELP* terpampang dimana-mana. Deskripsi dari penggunaan *command HELP* dapat dilihat pada tabel dibawah.

Karena takut command **HELP** ini disalahgunakan, Nimons memiliki ide untuk memberikan pesan yang mencegah terjadinya hal-hal yang tidak diinginkan dalam bentuk *footnote*.

```
# Kasus 1: belum dalam keadaan logged in
>>> HELP
```

```
===== HELP =====
```

Kamu belum login sebagai role apapun. Silahkan login terlebih dahulu.

1. LOGIN: Masuk ke dalam akun yang sudah terdaftar
2. REGISTER: Membuat akun baru

Footnote:

1. Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar
2. Jangan lupa untuk memasukkan input yang valid

```
# Kasus 2: Sudah login sebagai Dokter
>>> HELP
```

```
===== HELP =====
```

Halo Dokter [Neroifa](#). Kamu memanggil command HELP. Kamu pasti sedang kebingungan. Berikut adalah hal-hal yang dapat kamu lakukan sekarang:

1. LOGOUT: Keluar dari akun yang sedang digunakan
2. DIAGNOSIS: Melakukan diagnosis penyakit pasien berdasarkan kondisi tubuh pasien
3. # ...dan seterusnya

Footnote:

1. Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar
2. Jangan lupa untuk memasukkan input yang valid

```
# Kasus 3: Sudah login sebagai Pasien
>>> HELP
```

```
===== HELP =====
```

Selamat datang, [GRO](#). Kamu memanggil command HELP. Kamu pasti sedang kebingungan. Berikut adalah hal-hal yang dapat kamu lakukan sekarang:

1. LOGOUT: Keluar dari akun yang sedang digunakan
2. DAFTAR\_CHECKUP: Mendaftarkan diri untuk pemeriksaan dokter
3. # ...dan seterusnya

Footnote:

1. Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar
2. Jangan lupa untuk memasukkan input yang valid

```
# Kasus 4: Sudah login sebagai Manager
>>> HELP
```

```
===== HELP =====
```

Halo Manager [nimonsslatte](#). Kenapa kamu memanggil command HELP? Kan kamu manager, tapi yasudahlah kamu pasti sedang kebingungan. Berikut adalah hal-hal yang dapat kamu lakukan sekarang:

1. LOGOUT: Keluar dari akun yang sedang digunakan

2. TAMBAH\_DOKTER: Mendaftarkan dokter baru ke sistem
3. # ...dan seterusnya

Footnote:

1. Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar
2. Jangan lupa untuk memasukkan input yang valid

## F06 – Denah Rumah Sakit

### Spesifikasi Fitur (STI)

### Spesifikasi Fitur (IF & EL & EB)

## F07 – Lihat User

### Akses: Manager

Setelah berhasil login, Manager dapat melihat data seluruh pengguna (dokter dan pasien), atau secara spesifik hanya pasien atau hanya dokter. Manager dapat memilih metode pengurutan berdasarkan:

- ID (Numerik)
- Nama (Leksikografis, Case-Insensitive)

Tampilan kolom:

- LIHAT\_USER: ID, Nama, Role, dan Penyakit
- LIHAT\_PASIEN: ID, Nama, dan Penyakit
- LIHAT\_DOKTER: ID dan Nama

```
# DISCLAIMER: Tampilan/interface dibebaskan, berikut hanya contoh saja
# Kasus 1: Melihat data user (user bisa berarti dokter atau pasien)
>>> LIHAT_USER
Urutkan berdasarkan?
1. ID
2. Nama
>>> Pilihan: 1

Urutan sort?
1. ASC (A-Z)
2. DESC (Z-A)
>>> Pilihan: 1

Menampilkan semua pengguna dengan ID terurut ascending...
ID | Nama      | Role      | Penyakit
-----
1  | Jeffrey   | Dokter    | -
```

2	Erik	Pasien	Maag
3	Neroifa	Dokter	-
4	Daev	Pasien	Flu
5	Remon	Pasien	Maag
6	Alpin	Dokter	-

# Kasus 2: Spesifik melihat data pasien

>>> **LIHAT\_PASIEN**

Urutkan berdasarkan?

1. ID
2. Nama

>>> Pilihan: 2

Urutan sort ID?

1. ASC (A-Z)
2. DESC (Z-A)

>>> Pilihan: 2

Menampilkan pasien dengan nama terurut descending...

ID	Nama	Penyakit
5	Remon	Maag
2	Erik	Maag
4	Daev	Flu

-----

# Kasus 3: Spesifik melihat data dokter

>>> **LIHAT\_DOKTER**

Urutkan berdasarkan?

1. ID
2. Nama

>>> Pilihan: 2

Urutan sort?

1. ASC (A-Z)
2. DESC (Z-A)

>>> Pilihan: 1

Menampilkan dokter dengan nama terurut ascending...

ID	Nama
6	Alpin
1	Jeffrey
3	Neroifa

-----

## F08 – Cari User

### Akses: Manager

Manager dapat mencari data pengguna secara spesifik berdasarkan ID atau Nama melalui perintah (CARI\_USER), maupun secara lebih terfokus melalui (CARI\_PASIEN) dan (CARI\_DOKTER). Jika mencari berdasarkan **ID**, sistem akan menggunakan **algoritma binary search** karena data sudah diurutkan berdasarkan ID. Jika mencari berdasarkan **Nama**, sistem akan menggunakan **algoritma sequential search**. **ID dan Nama bersifat unik**, jadi

hasil pencarian hanya akan mengembalikan satu data atau tidak ditemukan sama sekali. Khusus untuk pencarian pasien berdasarkan penyakit, hasilnya bisa lebih dari satu data, satu data, atau tidak ada sama sekali.

```
# DISCLAIMER: Tampilan/interface dibebaskan, berikut hanya contoh saja
# Kasus 1: Mencari data user (user bisa berarti dokter atau pasien) berdasarkan ID
```

```
>>> CARI_USER
```

```
Cari berdasarkan?
```

1. ID
2. Nama

```
>>> Pilihan: 1
```

```
>>> Masukkan nomor ID user: 3
```

```
Menampilkan pengguna dengan nomor ID 3...
```

ID	Nama	Role	Penyakit
3	Neroifa	Dokter	-

```
# Kasus 2: User yang dicari tidak ditemukan
```

```
>>> CARI_USER
```

```
Cari berdasarkan?
```

1. ID
2. Nama

```
>>> Pilihan: 2
```

```
>>> Masukkan nama user: Kebin
```

```
Tidak ditemukan pengguna dengan nama Kebin!
```

```
# Kasus 3: Mencari data pasien
```

```
>>> CARI_PASIEN
```

```
Cari berdasarkan?
```

1. ID
2. Nama
3. Penyakit

```
>>> Pilihan: 3
```

```
>>> Masukkan nama penyakit: Maag
```

```
# Memungkinkan adanya pasien dengan penyakit Maag lebih dari satu
```

```
Urutkan berdasarkan?
```

1. ID
2. Nama

```
>>> Pilihan: 1
```

```
Urutan sort ID?
```

1. ASC (A-Z)
2. DESC (Z-A)

```
>>> Pilihan: 2
```

```
Menampilkan pasien dengan penyakit Maag dengan ID terurut descendant...
```

ID	Nama	Penyakit
5	Remon	Maag
2	Erik	Maag



```
# Kasus 4: Mencari data dokter
>>> CARI_DOKTER
Cari berdasarkan?
1. ID
2. Nama
>>> Pilihan: 2

>>> Masukkan nama dokter: Alpin

Menampilkan dokter dengan nama Alpin...
ID | Nama
-----
6  | Alpin
```

## F09 – Lihat Antrian

[Spesifikasi Fitur \(STI\)](#)

[Spesifikasi Fitur \(IF & EL & EB\)](#)

## F10 – Tambah Dokter

### Akses: Manager

Manager yang telah login bisa menambahkan dokter baru. Setelah itu Dokter baru bisa login dengan username dan password yang telah diberikan oleh manager. Username untuk dokter baru harus unik (silakan lakukan validasi dengan struktur data **set**).

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja

# Kasus 1: Dokter bernama Budi belum ada
>>> TAMBAH_DOKTER
Username: Budi
Password: budi123

Dokter Budi berhasil ditambahkan!
>>> LOGOUT

Sampai jumpa!

>>> LOGIN
Username: Budi
Password: budi123

Selamat pagi Dokter Budi!
```

```
# Kasus 2: Dokter bernama Budi sudah ada
>>> TAMBAH_DOKTER
Username: Denis
Password: denis123

Sudah ada Dokter bernama Denis!
```

Manager juga dapat melakukan assign ruangan ke dokter tertentu yang belum memiliki ruangan.

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja

# Kasus 1: Ruangan Kosong dan dokter belum di assign di ruang manapun
>>> ASSIGN_DOKTER
Username: Budi
Ruangan: A1

Dokter Budi berhasil diassign ke ruangan A1!

# Kasus 2: Ruangan Kosong dan dokter sudah di assign di ruang lain
>>> ASSIGN_DOKTER
Username: Budi
Ruangan: A2

Dokter Budi sudah diassign ke ruangan A1!

# Kasus 3: Ruangan tidak kosong dan dokter belum di assign di ruang manapun
>>> ASSIGN_DOKTER
Username: Adi
Ruangan: A1

Dokter Budi sudah menempati ruangan A1!
Silakan cari ruangan lain untuk dokter Adi.

# Kasus 4: Ruangan tidak kosong dan dokter sudah di assign di ruang lain
>>> ASSIGN_DOKTER
Username: Ahuy
Ruangan: A1

Dokter Ahuy sudah menempati ruangan A3!
Ruangan A1 juga sudah ditempati dokter Budi!
```

## F11 – Diagnosis

### Akses: Dokter

Dokter yang login bisa melakukan pengecekan penyakit terhadap pasien-pasiennya. Namun karena teknologi telah berkembang, dokter tidak perlu melakukan pengecekan secara manual terhadap pasien yang perlu diperiksa karena dokter saat ini memiliki teknologi "Auto-Diagnose-2.0". Teknologi ini bekerja dengan dengan membaca data kondisi pasien dan menyesuaikannya dengan batasan kondisi yang ada pada file [penyakit.csv](#).

#### NOTES:

- Khusus untuk **STI**, silakan buat penyimpanan data secara **hard coded**, namun tetap menyesuaikan dengan file yang diberi.
- Jika seorang pasien memenuhi kondisi beberapa penyakit, cukup pilih satu penyakit yang pertama kali muncul sesuai dengan urutan pada file [penyakit.csv](#).

```
# DISCLAIMER: Tampilan / interface dibebaskan, berikut hanya contoh saja
# KASUS 1: Dokter memiliki pasien yang perlu diperiksa
>>> DIAGNOSIS

Budi terdiagnosa penyakit Maag!

# KASUS 2: Antrian pasien sudah kosong dan tidak ada pasien yang perlu diperiksa
>>> DIAGNOSIS

Tidak ada pasien untuk diperiksa!

# KASUS 3: Pasien tidak terjangkau penyakit apapun setelah diperiksa. Jika terdapat kasus
ini maka pasien diperbolehkan pulang.
>>> DIAGNOSIS

Budi tidak terdiagnosis penyakit apapun!

# Kembali ke menu utama
```

## F12 – Ngobatin

#### Akses: Dokter

Setiap dokter seperti Dr. Neroifa, dilengkapi dengan sebuah teknologi inovatif "Auto-Prescription-2.0". Teknologi ini adalah sebuah sistem peresepan obat berbasis AI. Data train yang digunakan untuk melatih AI dapat dilihat pada file [obat\\_penyakit.csv](#). Setiap penyakit yang terdiagnosis dapat dipastikan memiliki list obat penyembuh yang berurutan. Tugas dokter adalah memberi obat kepada pasien secara terurut.

#### NOTES:

- Khusus untuk **STI**, silakan buat penyimpanan data secara **hard coded**, namun tetap menyesuaikan dengan file yang diberi.

Tips:

- Gunakan struktur data Map untuk penyimpanan penyakit - obat

```
# DISCLAIMER: Tampilan / interface dibebaskan, berikut hanya contoh saja

# KASUS 1: Pasien memiliki penyakit bernama "Maag"
>>> NGOBATIN

Dokter sedang mengobati pasien!
Pasien memiliki penyakit Maag
Obat yang harus diberikan:
1. Paracetamol
2. Oralit
3. Obat Maag
4. Paracetamol

# KASUS 2: Pasien tidak memiliki penyakit dan belum di diagnosis
>>> NGOBATIN

Dokter sedang mengobati pasien!
Pasien tidak memiliki penyakit!
Pasien belum di diagnosis!

# Kembali ke menu utama
```

## F13 – Aku boleh pulang ga, dok 🙄?

### Akses: Pasien

Setelah didiagnosis dan diberikan obat, pasien akan menjalani masa penyembuhan dengan mengonsumsi obat-obatan yang telah diresepkan. Setelah semua obat tersebut habis diminum, pasien dapat berkonsultasi kembali kepada dokter untuk menanyakan apakah kondisinya sudah cukup baik untuk diperbolehkan pulang. Idealnya, setiap pasien mengikuti petunjuk pengobatan yang telah dijelaskan oleh dokter. Namun, karena tidak ada pasien yang sempurna, penting bagi pasien untuk melakukan validasi atau pemeriksaan ulang kepada dokter untuk memastikan apakah mereka benar-benar sudah siap untuk pulang.

Dalam kasus dimana ada kesalahan urutan peminuman obat, maka pasien dapat meminum penawar untuk mengeluarkan obat yang telah diminumnya kembali ke *Inventory*.

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
```

# KASUS 1: Pasien belum diberikan diagnosa penyakit oleh dokter  
>>> PULANGDOK

Kamu belum menerima diagnosis apapun dari dokter, jangan buru-buru pulang!

# KASUS 2: Pasien belum menghabiskan seluruh obat yang diberikan kepadanya  
>>> PULANGDOK

Dokter sedang memeriksa keadaanmu...

Masih ada obat yang belum kamu habiskan, minum semuanya dulu yukk!

# KASUS 3: Pasien sudah menghabiskan obat, namun terdapat urutan yang salah dalam konsumsinya  
>>> PULANGDOK

Dokter sedang memeriksa keadaanmu...

Maaf, tapi kamu masih belum bisa pulang!

Urutan peminuman obat yang diharapkan:  
Paracetamol -> Oralit -> Paracetamol -> Aspirin

Urutan obat yang kamu minum  
Paracetamol -> Oralit -> Aspirin -> Paracetamol

Silahkan kunjungi dokter untuk meminta penawar yang sesuai !

# KASUS 4: Pasien sudah menghabiskan obat, dan semuanya valid.  
>>> PULANGDOK

Dokter sedang memeriksa keadaanmu...

Selamat! Kamu sudah dinyatakan sembuh oleh dokter. Silahkan pulang dan semoga sehat selalu!

# Kembali ke menu utama dan pasien meninggalkan ruangan.

## F14 - Daftar Check-Up

### Akses: Pasien

Pasien yang sudah login dapat mendaftar untuk melakukan check-up dengan dokter. Dalam proses pendaftaran, pasien diminta untuk memasukkan data medical check-up seperti tekanan darah, berat badan, dan tinggi badan. Setelah itu, pasien dapat memilih dokter yang tersedia dan akan dimasukkan ke dalam antrian dokter tersebut. Pasien juga dapat melihat informasi antrian mereka saat ini.

Secara garis besar:

1. Pendaftaran check-up dengan memasukkan data medis dasar
2. Pemilihan dokter dari daftar yang tersedia

### 3. Penempatan dalam antrian dokter

Notes:

- Gunakan struktur data **Map & Queue dengan representasi berkait** (Queue dengan Linked List)

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
```

```
# KASUS 1: Pasien berhasil mendaftar check-up
```

```
>>> DAFTAR_CHECKUP
```

```
Silakan masukkan data check-up Anda:
```

```
Suhu Tubuh (Celcius): 38.5
```

```
Tekanan Darah (sistol/diastol, contoh 120 80): 120 80
```

```
Detak Jantung (bpm): 100
```

```
Saturasi Oksigen (%): 98.5
```

```
Kadar Gula Darah (mg/dL): 80
```

```
Berat Badan (kg): 65.5
```

```
Tinggi Badan (cm): 175
```

```
Kadar Kolesterol (mg/dL): 250
```

```
Kadar Kolesterol LDL (mg/dL): 160
```

```
Trombosit (ribu/ $\mu$ L): 175000
```

```
Berikut adalah daftar dokter yang tersedia:
```

```
1. Dr. Budi - Spesialisasi Umum - Ruangan A1 (Antrian: 2 orang)
```

```
2. Dr. Cici - Spesialisasi Umum - Ruangan B3 (Antrian: 1 orang)
```

```
Pilih dokter (1-2): 1
```

```
Pendaftaran check-up berhasil!
```

```
Anda terdaftar pada antrian Dr. Budi di ruangan A1.
```

```
Posisi antrian Anda: 3
```

```
# KASUS 2: Pasien sudah terdaftar dalam antrian
```

```
>>> DAFTAR_CHECKUP
```

```
Anda sudah terdaftar dalam antrian check-up!
```

```
Silakan selesaikan check-up yang sudah terdaftar terlebih dahulu.
```

```
# KASUS 3: Input tidak valid
```

```
>>> DAFTAR_CHECKUP
```

```
Silakan masukkan data check-up Anda:
```

```
Suhu Tubuh (Celcius): -5
```

```
Suhu tubuh harus berupa angka positif!
```

```
Suhu Tubuh (Celcius): 38.5
```

```
Tekanan Darah (sistol diastol, contoh 120 80): -1 -1
```

```
Tekanan darah harus berupa angka positif!
```

```
Tekanan Darah (sistol diastol, contoh 120 80): 110 70
```

```
... (sesuaikan validasi dengan tipe data)
```

## F15 – Antrian Saya!

### Akses: Pasien

Pasien yang sudah melakukan pendaftaran dapat melihat status antriannya dengan perintah **ANTRIAN**.

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
```

```
# KASUS 1: Pasien sudah terdaftar dalam antrian
```

```
>>> ANTRIAN
```

```
Status antrian Anda:
```

```
Dokter: Dr. Budi
```

```
Ruangan: A1
```

```
Posisi antrian: 3 dari 4
```

```
# KASUS 2: Pasien belum terdaftar dalam antrian
```

```
>>> ANTRIAN
```

```
Anda belum terdaftar dalam antrian check-up!
```

```
Silakan daftar terlebih dahulu dengan command DAFTAR_CHECKUP.
```

## F16 – Minum Obat

### Akses: Pasien

Setelah diberikan obat oleh dokter dan menyimpannya di dalam *inventory*, Pasien ingin meminum obat tersebut. Untuk meminumnya, Pasien harus mengambil obat dari *inventory* dan memasukkannya ke dalam perut. Obat yang pertama diminum akan berada di bagian paling bawah perut.

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
```

```
>>> MINUM_OBAT
```

```
===== DAFTAR OBAT =====
```

```
1. Paracetamol
```

```
2. Amoxicillin
```

```
3. Panadol
```

```
# Kasus 1: Apabila obat yang dipilih tidak ada
```

```
>>> Pilih obat untuk diminum: 4
```

```
Pilihan nomor tidak tersedia!
```

```
# Kasus 2: Apabila obat yang dipilih ada
Pilih obat untuk diminum: 1

GLEKGLEKGLEK... Paracetamol berhasil diminum!!!
```

```
# Tampilan daftar obat setelah paracetamol diminum

>>> MINUM_OBAT

===== DAFTAR OBAT =====
1. Amoxicillin
2. Panadol

# dan seterusnya
```

## F17 – Minum Penawar

### Akses: Pasien

Suatu hari, seorang pasien merasa kesakitan karena salah minum obat yang diberikan dokter. Dokter segera memberikan penawar khusus untuk membantu pasien tersebut. Setelah pasien meminum penawar, obat terakhir yang sebelumnya diminum langsung dikeluarkan dari perut pasien dan kembali ke inventory obat.

Singkatnya:

Fitur "Minum Penawar" berfungsi untuk mengeluarkan obat terakhir yang diminum pasien dari stack perut dan mengembalikannya ke dalam inventory obat pasien.

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja

# Kasus 1: Obat terakhir yang diminum adalah Paracetamol
>>> PENAWAR
Uwekkk!!! Paracetamol keluar dan kembali ke inventory
```

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja

# Kasus 2: Belum ada obat yang diminum
>>> PENAWAR
Perut kosong!! Belum ada obat yang dimakan.
```



## F18 – Exit

Dokter, Pasien, dan Manager tidak dapat terus melakukan aktivitas mereka di rumah sakit. Mereka tetap perlu istirahat. Pada suatu saat, Dokter memutuskan untuk pensiun, Manager memutuskan untuk resign, dan pasien memutuskan untuk istirahat di rumah saja. Nimons mendengarkan permintaan mereka dan menyediakan suatu *command* lagi, yaitu *exit*. Namun, Nimons sudah siap melakukan dekonstruksi rumah sakit. Namun, karena Nimons baik hati, Nimons masih memberikan kesempatan bagi para Dokter, Pasien, dan Manager jika masih ingin melanjutkan karir mereka dan memperjuangkan kesehatan. Mereka diarahkan untuk melakukan *save* terlebih dahulu sebelum rumah sakit dihancurkan.

```
>>> EXIT
```

```
Apakah Anda mau melakukan penyimpanan file yang sudah diubah? (y/n) n
```

```
# Keluar program
```

```
>>> EXIT
```

```
# Contoh input tidak valid
```

```
Apakah Anda mau melakukan penyimpanan file yang sudah diubah? (y/n) a
```

```
Apakah Anda mau melakukan penyimpanan file yang sudah diubah? (y/n) Y
```

```
# Menjalankan prosedur save (F15) dan keluar program
```

### NOTES:

Khusus untuk **STI**, tidak perlu membuat logic penyimpanan file, cukup berikan pesan ketika keluar program.

# Spesifikasi Program Tambahan (STI)

## S01 – Denah Rumah Sakit

### Akses: Manager, Dokter, Pasien

Seluruh role (Manajer, Pasien, dan Dokter) dapat melihat denah ruangan di rumah sakit serta detail dari ruangan tersebut, berikut merupakan gambaran command yang dapat dilakukan. Ukuran dari denah rumah sakit dimasukan kedalam program dengan bantuan [argumen](#). Pengguna juga dapat melihat ruangan dengan detail informasi yang ditampilkan pada contoh.

Notes:

- Denah akan berbentuk ruangan yang memanjang (gunakan ADT List)

```
>>> LIHAT_DENAH
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
# Kasus 1: ruangan terdapat dokter dan pasien
```

```
>>> LIHAT_RUANGAN 1
```

```
--- Detail Ruangan 1 ---
```

```
Kapasitas : 3
```

```
Dokter : Dr. Strange
```

```
Pasien di dalam ruangan:
```

```
1. John Smith
```

```
2. John Lennon
```

```
-----
```

```
# Kasus 2: ruangan terdapat dokter dan tidak ada pasien
```

```
>>> LIHAT_RUANGAN 2
```

```
--- Detail Ruangan 2 ---
```

```
Kapasitas : 3
```

```
Dokter : Dr. Oz
```

```
Pasien di dalam ruangan:
```

```
Tidak ada pasien di dalam ruangan saat ini.
```

```
-----
```

```
# Kasus 3: ruangan tidak terdapat dokter.
```

```
>>> LIHAT_RUANGAN 3
```

```
--- Detail Ruangan 3 ---
```

```
Kapasitas : 3
```

```
Dokter : -
```

```
Pasien di dalam ruangan:  
  Tidak ada pasien di dalam ruangan saat ini.  
-----
```

## S02 – Lihat Antrian

### Akses: Manager

Manager dapat melihat rincian di seluruh ruangan saat ini. Ruangan yang perlu ditampilkan adalah ruangan yang tidak kosong, dalam arti ruangan yang memiliki dokter didalamnya. Informasi yang harus ditampilkan kurang lebih meliputi denah diikuti dengan detail ruang yang tidak kosong beserta antrian.

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja  
>>> LIHAT_SEMUA_ANTRIAN
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
===== 1 =====
```

```
Kapasitas : 3  
Dokter    : Dr. Strange  
Pasien di dalam ruangan:  
  1. John Smith  
  2. John Lennon  
  3. John Lagi  
Pasien di antrian:  
  1. John Terus  
  2. John Bosen
```

```
# Ruangan Kosong tidak perlu ditampilkan
```

```
===== 2 =====
```

```
Kapasitas : 3  
Dokter    : Dr. Oz  
Pasien di dalam ruangan:  
  Tidak ada pasien di dalam ruangan saat ini.  
Pasien di antrian:  
  Tidak ada pasien di antrian saat ini.
```

# Spesifikasi Program Tambahan (IF & EL & EB)

## D01 – Denah Rumah Sakit

### Akses: Manager, Dokter, Pasien

Seluruh role (Manajer, Pasien, dan Dokter) dapat melihat denah ruangan di rumah sakit serta detail dari ruangan tersebut, berikut merupakan gambaran command yang dapat dilakukan. Ukuran dari denah rumah sakit terdapat dalam file konfigurasi. Pengguna juga dapat melihat ruangan dengan detail informasi yang ditampilkan pada contoh.

```
>>> LIHAT_DENAH
```

```
      1      2      3      4
+-----+-----+-----+
A | A1 | A2 | A3 | A4 |
+-----+-----+-----+
B | B1 | B2 | B3 | B4 |
+-----+-----+-----+
C | C1 | C2 | C3 | C4 |
+-----+-----+-----+
D | D1 | D2 | D3 | D4 |
+-----+-----+-----+
```

```
# Kasus 1: ruangan terdapat dokter dan pasien
```

```
>>> LIHAT_RUANGAN A1
```

```
--- Detail Ruangan A1 ---
```

```
Kapasitas : 3
```

```
Dokter : Dr. Strange
```

```
Pasien di dalam ruangan:
```

```
1. John Smith
```

```
2. John Lennon
```

```
-----
```

```
# Kasus 2: ruangan terdapat dokter dan tidak ada pasien
```

```
>>> LIHAT_RUANGAN B1
```

```
--- Detail Ruangan B1 ---
```

```
Kapasitas : 3
```

```
Dokter : Dr. Oz
```

```
Pasien di dalam ruangan:
```

```
Tidak ada pasien di dalam ruangan saat ini.
```

```
-----
```

```
# Kasus 3: ruangan tidak terdapat dokter.
```

```
>>> LIHAT_RUANGAN C3
```

```
--- Detail Ruangan C3 ---
```

```
Kapasitas : 3
```

```
Dokter : -
```

```
Pasien di dalam ruangan:
```

Tidak ada pasien di dalam ruangan saat ini.  
-----

## D02 – Lihat Antrian

### Akses: Manager

Manager dapat melihat rincian di seluruh ruangan saat ini. Ruangan yang perlu ditampilkan adalah ruangan yang tidak kosong, dalam arti ruangan yang memiliki dokter didalamnya. Informasi yang harus ditampilkan kurang lebih meliputi denah diikuti dengan detail ruang yang tidak kosong beserta antrian.

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
```

```
>>> LIHAT_SEMUA_ANTRIAN
```

	1	2	3	4
A	A1	A2	A3	A4
B	B1	B2	B3	B4
C	C1	C2	C3	C4
D	D1	D2	D3	D4

```
===== A1 =====
```

```
Kapasitas : 3
```

```
Dokter : Dr. Strange
```

```
Pasien di dalam ruangan:
```

1. John Smith
2. John Lennon
3. John Lagi

```
Pasien di antrian:
```

1. John Terus
2. John Bosen

```
# Ruangan Kosong tidak perlu ditampilkan
```

```
===== B1 =====
```

```
Kapasitas : 3
```

```
Dokter : Dr. Oz
```

```
Pasien di dalam ruangan:
```

```
Tidak ada pasien di dalam ruangan saat ini.
```

```
Pasien di antrian:
```

```
Tidak ada pasien di antrian saat ini.
```

## D03 – Load

Setelah menjalani masa istirahat akibat tingginya aktivitas dan volume pekerjaan, rumah sakit ingin kembali beroperasi seperti semula dan melanjutkan kegiatan yang sempat tertunda. Oleh karena itu, rumah sakit melakukan *load*.

Prosedur ini akan memuat data yang sesuai dengan [struktur data eksternal](#). Prosedur ini akan dijalankan sekali ketika pengguna memulai program. Cara menjalankan prosedur ini adalah dengan memberikan **nama folder** yang berisi file penyimpanan.

Untuk prosedur ini, asumsikan bahwa seluruh file penyimpanan dalam suatu folder :

1. Terjamin ada
  - a. Untuk **user.csv** dipastikan telah memiliki setidaknya satu akun manajer.
  - b. Tidak mungkin ada kasus **config.txt** tidak terdapat dalam folder. Jadi, program rumah sakit hanya dapat berjalan ketika semua file terdefinisi.
2. Memiliki nama yang *fixed*
3. Memiliki format yang sesuai dengan struktur data eksternal. Akan, tetapi bentuk struktur data pada program dibebaskan, silakan definisikan sendiri struktur data terbaik yang dapat digunakan pada program.

Namun, prosedur tetap harus melakukan validasi apakah folder tersebut **ada atau tidak**.

Program hanya akan melakukan read pada file CSV apabila pengguna menjalankan fungsi load.

**Hint:** Gunakan argumen untuk menerima argumen saat menjalankan file executable.

[Tutorial argumen](#).

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
~$ ./main <<nama_folder>>
# parent folder dari nama_folder akan sama seperti dalam fungsi save

Loading...
# panggil prosedur load data
Selamat datang kembali di rumah sakit Nimons !
# meminta perintah berikutnya... (cth : register, login, dll)

# user tidak memberikan nama folder
~$ ./main
```

```
Tidak ada nama folder yang diberikan!  
Usage : ./main <<nama_folder>>  
# program keluar
```

```
# user memberikan folder yang tidak ada  
~$ ./main folder_palsu  
  
Folder "folder_palsu" tidak ditemukan.  
# program keluar
```

## D04 – Save

Prosedur ini digunakan untuk menyimpan data ke dalam file yang sesuai dengan [struktur data eksternal](#). Dalam prosedur ini, program akan meminta **nama folder** yang akan digunakan sebagai tempat penyimpanan file. Berikut adalah ketentuannya:

1. Jika nama folder tidak ditemukan, program akan membuat folder sesuai dengan masukan dan memberikan pesan bahwa ia **membuat folder baru** dan **berhasil menyimpan data** di folder tersebut.
2. Jika nama folder ditemukan namun belum ada file pada folder tersebut maka program akan **menaruh file baru** pada folder tersebut
3. Jika nama folder dan file sudah ada, program akan mengganti file pada folder tersebut dengan yang lebih baru dan tidak perlu memberikan pesan tambahan. (*overwrite/replace*)
- 4.

### Note :

1. Boleh menentukan folder *parent save*, misalnya untuk contoh di bawah "data/"
2. Manipulasi yang dilakukan terhadap data program akan disimpan terlebih dahulu dalam suatu variabel (in memory). Program hanya akan melakukan write terhadap file csv ataupun txt ketika pengguna menjalankan fungsi save.

```
# nama folder tidak ditemukan, folder belum dibuat  
>>> SAVE  
  
Masukkan nama folder: 09-03-2024  
# Folder 09-03-2024 belum ada  
  
Saving...  
  
Membuat folder data/09-03-2024...  
Berhasil menyimpan data di folder data/09-03-2024!
```

```
# nama folder ditemukan
```

```
>>> SAVE
```

```
Masukkan nama folder: 09-03-2024
```

```
# Folder 09-03-2024 sudah ada
```

```
Saving...
```

```
Berhasil menyimpan data di folder data/09-03-2024!
```

```
# apabila program melakukan overwrite/replace, tidak diperlukan pesan tambahan
```

```
>>> SAVE
```

```
Masukkan nama folder: 09-03-2024
```

```
# Folder data/ belum ada
```

```
Saving...
```

```
Membuat folder data...
```

```
Membuat folder data/09-03-2024...
```

```
Berhasil menyimpan data di folder data/09-03-2024!
```




# Spesifikasi Bonus

Karena beberapa Nimons menyukai tantangan, terdapat beberapa Spesifikasi Bonus yang dapat dikerjakan untuk menambah kreativitas dari program. Perlu dicatat untuk mengutamakan pengerjaan Spesifikasi Utama terlebih dahulu, karena nilai sempurna juga dapat diraih dengan program yang dapat memenuhi seluruh Spesifikasi Utama dengan sempurna. Kerjakan Spesifikasi Bonus jika merasa Spesifikasi Utama sudah selesai dengan baik serta masih memiliki waktu lebih.

## B01 – Git Best Practice

Penggunaan Git secara spesifikasi wajib, hanyalah sebagai wadah pengumpulan. Satu kali upload file / commit yang berisi seluruh program, lalu membuat Release Tag sudah dianggap cukup. Pada bonus ini, kalian harus menerapkan penggunaan Git secara bersamaan dengan anggota kalian dan menerapkan *best practice* yang ada, yaitu melingkupi Commit Message dan Branching. Video tutorial Git dapat dilihat pada tautan berikut:

 Basic Git Tutorial.mp4

### Commit Message

Commit message pada git digunakan sebagai log untuk menjelaskan perubahan yang terjadi pada repository. Commit message yang baik harus jelas, ringkas, dan informatif. Karakteristik commit message yang baik:

**Concise:** Singkat dan deskriptif


**Relevance:** Fokus terhadap kenapa perubahan tersebut dilakukan, bukan bagaimana

**Clarity:** Hindari deskripsi yang ambigu, pesan harus spesifik


**Consistency:** Pilih dan ikuti format standar (bisa dicari di internet) sehingga mudah untuk dibaca

Commit message yang baik

**fix: fix property value display behaviour when editing and selected tool is cursor**

 ditramadia committed 2 weeks ago

**fix: set width and height rectangle**

 bernarduswillson committed 2 weeks ago

Commit message yang jahat



fix sama gasssss ini maksudnya apaaa???

Contoh standar commit message:

<type>: <description>

type:

**feat** - Penambahan fitur baru

**fix** - Bug fix

**docs** - Perubahan pada dokumentasi (README, laporan, dsb)

**style** - Formatting, penambahan type safe, dsb

**refactor** - Refactor kode, ganti nama variable, dsb

**test** - Penambahan/perubahan script untuk testing

**chore** - Perubahan pada config file

style: add type safe for RNG algorithm

feat: add RNG algorithm

feat: add database for monsters

refactor: refactor command parser

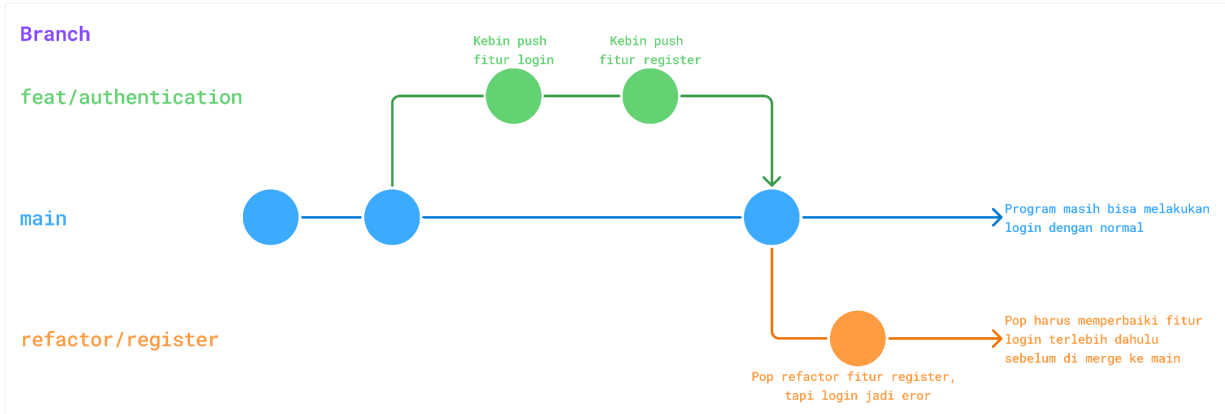
feat: add command parser

## Branching

Branching adalah pembagian cabang dari repository kalian. Branching berfungsi untuk memisahkan suatu pekerjaan dengan pekerjaan lainnya (isolasi). Dengan branching, kalian bisa bekerja secara paralel pada branch yang berbeda untuk memitigasi *conflict*.

Misal Kebin dan Pop tidak menerapkan branching dan hanya bekerja pada branch utama (branch **master** atau **main**). Kebin membuat fitur Login dan Register lalu di-push. Kemudian, Pop melakukan refactor fitur Register lalu di-push. Walaupun fitur Register tetap bekerja, ternyata fitur Login jadi error. Pada akhirnya Kebin tidak bisa melakukan login dan tidak bisa melanjutkan pekerjaannya sebelum fitur Login diperbaiki.

Alternatifnya, Kebin dan Pop menerapkan branching. Kebin telah membuat fitur Login dan Register di merge terpisah, lalu di-merge ke branch utama. Pop membuat branch baru terlebih dahulu, sebelum mengubah fitur Register dan melakukan push. Saat fungsi Register telah di-refactor, ternyata fitur Login jadi error. Tetapi, karena Pop bekerja pada branch terpisah, kode Pop tidak berpengaruh pada kode Kebin. Pop harus memperbaiki fitur Login terlebih dahulu, dalam waktu yang bersamaan, Kebin tetap dapat melakukan Login dan melanjutkan pekerjaannya.



## B02 – Denah Dinamis

### Akses: Manager

Fitur ini memungkinkan Manager untuk mengubah tata letak denah rumah sakit secara dinamis. Ukuran denah rumah sakit (jumlah baris dan kolom) tidak lagi bersifat tetap saat program dimulai, melainkan dapat diubah selama program berjalan. Selain mengubah dimensi keseluruhan, fitur ini juga mencakup kemampuan untuk memindahkan dokter antar ruangan yang tersedia.

Notes:

- Khusus untuk STI, bonus implementasi denah dinamis termasuk ke perubahan struktur denah dari satu baris menjadi bentuk matrik (seperti spesifikasi IF & EL & EB)..

```
# Denah awal berukuran 2x3
>>> LIHAT_DENAH
```

```

      1      2      3
+-----+-----+
A | A1 | A2 | A3 |
+-----+-----+
B | B1 | B2 | B3 |

```

```
+-----+-----+-----+

# Manajer ingin mengubah ukuran denah
>>> UBAH_DENAH 3 4
Denah rumah sakit berhasil diubah menjadi 3 baris dan 4 kolom.

>>> LIHAT_DENAH

      1      2      3      4
+-----+-----+-----+
A | A1 | A2 | A3 | A4 |
+-----+-----+-----+
B | B1 | B2 | B3 | B4 |
+-----+-----+-----+
C | C1 | C2 | C3 | C4 |
+-----+-----+-----+

# Manager mencoba mengecilkan denah ke 1x1, tapi masih terdapat dokter di A2 dan A3
>>> UBAH_DENAH 1 1
Tidak dapat mengubah ukuran denah. Ruang A2 masih ditempati oleh Dr. Strange. Silakan
pindahkan dokter terlebih dahulu.

>>> LIHAT_DENAH

      1      2      3      4
+-----+-----+-----+
A | A1 | A2 | A3 | A4 |
+-----+-----+-----+
B | B1 | B2 | B3 | B4 |
+-----+-----+-----+
C | C1 | C2 | C3 | C4 |
+-----+-----+-----+

# Kasus pindah dokter berhasil
>>> PINDAH_DOKTER A2 C4
Dr. Strange berhasil dipindahkan dari ruangan A2 ke ruangan C4.

# Kasus ruangan tujuan sudah ditempati
>>> PINDAH_DOKTER A3 C4
Pemindahan gagal. Ruang C4 Sudah ditempati.

# Kasus ruangan asal kosong
>>> PINDAH_DOKTER B4 A1
Pemindahan gagal. Ruang B4 Kosong.
```

## B03 – Aura!

Setelah beberapa kali terjadi malpraktik, sejumlah pasien di Nimons mulai memperhatikan jumlah *aura* dari dokter yang akan merawat mereka. Untuk itu, setiap dokter memiliki atribut *aura* yang dicatat dalam sistem.

Pada awalnya, aura setiap dokter bernilai 0. Setelah pasien berhasil diobati dan diperbolehkan pulang, aura dari dokter yang mengobatinya akan bertambah sebesar 1. Berikut adalah ilustrasinya:

#### Akses: Pasien

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
>>> PULANGDOK

Dokter sedang memeriksa keadaanmu...

Selamat! Kamu sudah dinyatakan sembuh oleh dokter. Silahkan pulang dan semoga sehat selalu!

# Nilai aura dokter yang merawat pasien saat ini bertambah sebesar 1
# Kembali ke menu utama dan pasien meninggalkan ruangan.
```

Selain itu, terdapat beberapa penyesuaian pada fungsionalitas F07 dan F13:

#### - F07 - Lihat User

Saat mengakses daftar dokter, manager dapat memilih untuk mengurutkan berdasarkan aura. Selain itu, nilai aura dari setiap dokter juga ditampilkan. Berikut adalah ilustrasinya:

#### Akses: Manager

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
>>> LIHAT_DOKTER
Urutkan berdasarkan?
1. ID
2. Nama
3. Aura
>>> Pilihan: 3

Urutan sort?
1. ASC (A-Z)
2. DESC (Z-A)
>>> Pilihan: 1

Menampilkan dokter dengan aura terurut ascending...
ID | Nama      | Aura
-----
6  | Alpin      | 1
1  | Jeffrey    | 6
3  | Neroifa    | 10
```

#### - F13 - Daftar Check-up

Saat mendaftar check-up, sistem akan menampilkan daftar dokter yang tersedia beserta jumlah aura yang mereka miliki. Berikut adalah ilustrasinya:

### Akses: Pasien

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
>>> DAFTAR_CHECKUP

Silakan masukkan data check-up Anda:
Tekanan Darah (sistol/diastol, contoh 120 80): 120 80
Berat Badan (kg): 65
Tinggi Badan (cm): 170
Keluhan: Sakit kepala dan demam selama 2 hari

Berikut adalah daftar dokter yang tersedia:
1. Dr. Budi - Spesialisasi Umum - Ruangan A1 (Antrian: 2 orang) - Aura 5
2. Dr. Cici - Spesialisasi Umum - Ruangan B3 (Antrian: 1 orang) - Aura 2

Pilih dokter (1-2): 1

Pendaftaran check-up berhasil!
Anda terdaftar pada antrian Dr. Budi di ruangan A1.
Posisi antrian Anda: 3
```

### NOTES:

- Sesuaikan atribut aura dokter pada struktur data file user.csv

## B04 – Banarich!!!!!! 🍌

Karena operasional rumah sakit juga ternyata membutuhkan uang yang cukup banyak. Pak Gro akhirnya memiliki ide menarik untuk menciptakan sistem currency di rumah sakit tersebut. Pak Gro ingin setiap pasien yang berobat harus membayar sejumlah Banarich kepada dokter, dan setiap keuntungan dokter harus disisihkan sekitar **20%** untuk biaya operasional rumah sakit. Oleh karena itu, akhirnya dokter memasang tarif berobat.

### Notes:

- Apabila 20% dari biaya yang ditarik kurang dari satu, maka dokter harus tetap memberikan minimal satu Banarich

### Akses: Pasien

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja

# Pre Kondisi:
# - Pasien memiliki Banarich sebanyak 10

# Kasus 1: Banarich tidak cukup
>>> DAFTAR_CHECKUP

Silakan masukkan data check-up Anda:
Tekanan Darah (sistol/diastol, contoh 120 80): 120 80
```

Berat Badan (kg): 65  
Tinggi Badan (cm): 170  
Keluhan: **Sakit kepala dan demam selama 2 hari**

Berikut adalah daftar dokter yang tersedia:

1. Dr. Budi - Spesialisasi Umum - Ruangan A1 (Antrian: 2 orang) - Aura 5 - Biaya 5
2. Dr. Cici - Spesialisasi Umum - Ruangan B3 (Antrian: 1 orang) - Aura 2 - Biaya 3
3. Dr. Doni - Spesialisasi Umum - Ruangan B2 (Antrian: 10 orang) - Aura 999 - Biaya 100

Pilih dokter (1-3): 3

Maaf Banarich kamu tidak cukup!  
Silakan pilih dokter lain atau mainkan mesin gacha!

# Kasus 2: Banarich cukup  
>>> **DAFTAR\_CHECKUP**

Silakan masukkan data check-up Anda:  
Tekanan Darah (sistol/diastol, contoh 120 80): **120 80**  
Berat Badan (kg): **65**  
Tinggi Badan (cm): **170**  
Keluhan: **Sakit kepala dan demam selama 2 hari**

Berikut adalah daftar dokter yang tersedia:

1. Dr. Budi - Spesialisasi Umum - Ruangan A1 (Antrian: 2 orang) - Aura 5 - Biaya 5
2. Dr. Cici - Spesialisasi Umum - Ruangan B3 (Antrian: 1 orang) - Aura 2 - Biaya 3
3. Dr. Doni - Spesialisasi Umum - Ruangan B2 (Antrian: 10 orang) - Aura 999 - Biaya 100

Pilih dokter (1-3): 1

# Banarich milik pasien akan berkurang sebanyak 5 Banarich  
# Pihak rumah sakit akan mendapatkan Banarich sebanyak 20% dari biaya, yaitu 1 Banarich  
# Banarich yang didapatkan oleh dokter adalah sisa biaya setelah dikurangi biaya operasional, yaitu 4 Banarich

Pendaftaran check-up berhasil!  
Anda terdaftar pada antrian Dr. Budi di ruangan A1.  
Posisi antrian Anda: 3

Karena sistem currency mulai diterapkan, para dokter dan pasien kini memiliki kemampuan tambahan untuk melihat jumlah Banarich yang dimilikinya saat ini.

### Akses: Pasien dan Dokter

# **DISCLAIMER:** Tampilan/ interface dibebaskan, berikut hanya contoh saja  
>>> **LIHAT\_DOMPET**  
Banarich milik kamu saat ini adalah 10 Banarich.

Pihak Manager rumah sakit juga dapat melihat kondisi keuangan rumah sakit saat ini.

### Akses: Manager

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
```

```
>>> LIHAT_FINANSIAL
```

```
Banarich milik rumah sakit saat ini adalah 1000 Banarich.
```

Para Pasien ternyata tidak setuju dengan keputusan sepihak yang diputuskan oleh Pak Gro. Mereka protes dan melontarkan pertanyaan “kalau misal Banarich mereka habis, lalu bagaimana cara mereka bisa berobat?!”. Pak Gro yang ketakutan pun akhirnya memberikan solusi kepada pasien, yaitu dengan mengadakan mesin “Gacha X 2.0 Alpha”. Mesin tersebut dapat memberikan pasien uang yang random jumlahnya.

Notes:

- Jumlah Banarich random yang dikeluarkan tidak boleh menggunakan library tambahan, harus diimplementasikan dengan menggunakan [RNG \(dengan LCG\)](#).
- Batas maksimum dari Banarich yang dapat dikeluarkan dibebaskan, selama masih masuk akal.

#### Akses: Pasien

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
```

```
>>> GACHA_GAMING
```

```
Mesin Gacha X 2.0 Alpha mengeluarkan 5 Banarich.
```

## B05 – Dead or Alive?!

Setelah beberapa kali pasien Nimons salah minum obat, Pak Gro menyadari kondisi pasien makin kritis. Melihat hal ini, Nimons Lab menciptakan aturan baru:

Setiap pasien Nimons hanya diberi 3 kesempatan kesalahan minum obat. Setiap kali pasien minum obat yang salah, nyawa mereka berkurang satu (●). Jika pasien mencapai 3 kesalahan atau kondisi kritis tertentu, pasien akan dianggap "tidak tertolong" alias game over, dan akan dikeluarkan dari sistem rumah sakit Nimons.

 Aturan Singkat:

- Setiap pasien memulai dengan 3 nyawa (●●●).
- Setiap salah minum obat, nyawa pasien berkurang 1.



- Jika nyawa habis (0), pasien dinyatakan "Ded" dan harus dikeluarkan dari rumah sakit.
- Pasien bisa sembuh dengan meminum obat dalam urutan yang benar atau menggunakan penawar sebelum mencapai 3 kesalahan.
- Tiap kali fitur Penawar digunakan, sisa nyawa wajib ditampilkan.
- Tampilan/Interface dibebaskan

#### Contoh Kasus:

- Stewart salah minum obat( minta Penawar ): ●●● (sisa 2 nyawa)
- Stewart salah minum obat lagi( minta Penawar lagi ): ●●● (sisa 1 nyawa)
- Pasien melakukan kesalahan minum obat( minta Penawar ) sebanyak 3 kali: ●●● (Kritis!) Kondisi pasien menjadi fatal, sehingga pasien tersebut dinyatakan "ded" dan otomatis tidak bisa login kembali (data pasien dihapus dari sistem).

## B06 – Mainin Antrian

**Akses:** Pasien

### Deskripsi Singkat:

Fitur ini memungkinkan pasien yang sedang berada dalam antrian dokter untuk melakukan dua aksi khusus: maju langsung ke posisi terdepan (SKIP\_ANTRIAN) atau membatalkan posisi antriannya (CANCEL\_ANTRIAN). Kedua aksi ini memerlukan validasi untuk memastikan pasien memang sedang berada dalam sebuah antrian. Aksi SKIP\_ANTRIAN juga memiliki validasi tambahan jika pasien sudah berada di posisi terdepan.

### Command: SKIP\_ANTRIAN

Pasien yang sedang mengantri dapat menggunakan command ini untuk langsung pindah ke posisi paling depan (posisi 1) di antrian dokter yang sedang diikutinya.

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja

# KASUS 1: Pasien (misal: GRO) sedang dalam antrian Dr. Budi (posisi 3 dari 4) dan ingin skip
# Kondisi Awal (dilihat dari F14 - ANTRIAN):
# Dokter: Dr. Budi
# Ruangan: A1
# Posisi antrian: 3 dari 4

>>> SKIP_ANTRIAN

Anda berhasil maju ke depan antrian Dr. Budi di ruangan A1!
Posisi antrian Anda sekarang: 1
```

```
# Kondisi Akhir (dilihat dari F14 - ANTRIAN):  
# Dokter: Dr. Budi  
# Ruangan: A1  
# Posisi antrian: 1 dari 4 (Pasien yang tadinya di posisi 1 dan 2 bergeser ke 2 dan 3)
```

```
# KASUS 2: Pasien (misal: GRO) mencoba skip padahal sudah di depan  
# Kondisi Awal (dilihat dari F14 - ANTRIAN):  
# Dokter: Dr. Budi  
# Ruangan: A1  
# Posisi antrian: 1 dari 4
```

**>>> SKIP\_ANTRIAN**

Anda sudah berada di posisi paling depan antrian! Tidak bisa skip lagi.

```
# KASUS 3: Pasien (misal: GRO) mencoba skip padahal tidak sedang dalam antrian  
# Kondisi Awal: Pasien GRO tidak terdaftar di antrian manapun
```

**>>> SKIP\_ANTRIAN**

Skip antrian gagal! Anda tidak sedang terdaftar dalam antrian manapun!

### **Command: CANCEL\_ANTRIAN**

Pasien yang sedang mengantri dapat menggunakan command ini untuk keluar sepenuhnya dari antrian dokter yang sedang diikutinya.

```
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja  
# KASUS 1: Pasien (misal: GRO) sedang dalam antrian Dr. Budi (posisi 1 dari 4) dan ingin cancel  
# Kondisi Awal (dilihat dari F14 - ANTRIAN):  
# Dokter: Dr. Budi  
# Ruangan: A1  
# Posisi antrian: 1 dari 4
```

**>>> CANCEL\_ANTRIAN**

Anda berhasil keluar dari antrian Dr. Budi di ruangan A1.

```
# Kondisi Akhir: Pasien GRO tidak lagi terdaftar di antrian manapun.  
# Antrian Dr. Budi kini memiliki 3 orang, pasien yang tadinya di posisi 2, 3, 4 kini menjadi 1, 2, 3.
```

```
# KASUS 2: Pasien (misal: GRO) mencoba cancel padahal tidak sedang dalam antrian  
# Kondisi Awal: Pasien GRO tidak terdaftar di antrian manapun
```

**>>> CANCEL\_ANTRIAN**

Pembatalan antrian gagal! Anda tidak sedang terdaftar dalam antrian manapun!

## BXX – Kreativitas

Pak Gro sebagai stakeholder tertinggi rumah sakit sangat menyukai Nimons yang kreatif dan dapat berpikir *out of the box*. Oleh karena itu, kalian dapat mengimplementasikan **fitur di luar yang telah disebutkan** di spesifikasi. Sebelum itu, silakan mengajukan fitur tersebut kepada Asisten, atau via Sheets QNA. Perlu dicatat bahwa fitur tambahan tersebut jangan sampai merubah alur dari spesifikasi utama. Jika argumen dan fitur yang kalian ajukan cukup mengesankan, kalian akan diberi **nilai bonus** oleh Pak Gro.

Berikut adalah beberapa contoh fitur bonus lainnya yang dapat diimplementasikan:

- Kreativitas penggunaan ASCII art ataupun memberikan warna text output untuk mempercantik tampilan program.
- Melakukan Enkripsi terhadap penyimpanan password dalam database menggunakan algoritma Caesar Cipher atau algoritma lainnya yang dibuat sendiri. Buatlah fungsi untuk melakukan *encrypt* serta *decrypt*.
- Membuat sistem keuangan pasien dan dokter, serta menampilkan leaderboard dokter terkaya saat program akan berhenti.
- Dan lain-lain....

# Struktur Data File

Program perlu membaca beberapa data dari file eksternal untuk mengoperasikan sistem ini. Format file yang diminta adalah file dengan ekstensi .csv yang dipisahkan dengan semicolon/titik koma (;)!

Namun sayangnya, **Dr. Naik Vario**, mantan dokter RS Nimons yang kini menjadi buronan karena kelakuannya yang suka "ngoprek sistem", melakukan **cyber attack** terhadap server pusat rumah sakit. Akibatnya, **fungsi-fungsi pembaca file CSV bawaan di bahasa C seperti fscanf, strtok, maupun sscanf jadi tidak bisa digunakan secara langsung untuk mem-parsing data!** 🤯

Akibat kerusakan tersebut, seluruh data pasien, penyakit, dan obat yang sebelumnya disimpan dalam format .csv **tidak bisa lagi dibaca otomatis oleh sistem**. Ini menyebabkan **dokter kebingungan, perawat panik, dan Gro mulai teriak-teriak di lorong rumah sakit** karena jadwal operasi jadi kacau.

Oleh karena itu, **Agen Stewart**, Nimon paling jenius dalam dunia pointer dan memory address, diberi misi penting: **membuat ulang parser CSV dari nol, tanpa fungsi strtok, strsep, atau sscanf**, hanya menggunakan **fungsi dasar seperti fgetc, fgets, dan putchar jika diperlukan**.

Berikut adalah contoh isi file-csv:

```
Id;username;password;role;penyakit;tekanan_darah;berat_badan;tinggi_badan
1;nimonsslatte;nimonatutgajah23;manager;;;
2;Neroifa;Neroifa123;dokter;maag;120/80;65;170
3;GRO;NeirofaCantik123;pasien;;;
```

Silakan membuat CSV untuk file lainnya sendiri. CSV akan disimpan pada sebuah folder yang sudah didefinisikan pada *command* save, dan nama file bersifat sudah pasti/*fixed* untuk memudahkan pengerjaan. Untuk template CSV dapat diakses pada tautan berikut:

[IF1210 Sample Database / Config Tugas Besar](#)

Selain itu, terdapat konfigurasi *state* rumah sakit yang terakhir kali disimpan dan disimpan dengan nama file **config.txt** mempunyai format seperti berikut.

Line 1	2 3
.	3
.	10 2 3 1
.	11 4 5

.	12 6
.	13 8 7
.	0
.	15 0
.	2
.	3 1 2 3 4
Line 11	5 2 3 4

Keterangan:

- a. Baris pertama (2 3) - berarti ukuran denah rumah sakit adalah 2x3, dengan panjang 2 dan lebar 3.
- b. Baris kedua (3) - berarti jumlah pasien maksimal untuk setiap ruangan adalah sebanyak 3 pasien.
- c. Baris ketiga hingga kedelapan untuk menandakan keadaan *state* antrian ketika disimpan.
  - i. Contoh, baris ketiga (10 2 3 1), karena hitungannya indeks 0 (identik ruangan A1), terdapat dokter dengan id 10, dan pasien-pasiennya dengan id 2, 3, dan 1.
  - ii. Berlaku selanjutnya hingga baris kedelapan yang berarti indeks 5 (identik ruangan B3), dengan dokter id 15, namun tanpa pasien, karena (iii).
  - iii. 0 berarti di ruangan tersebut tidak terdapat pasien sama sekali.
  - iv. Catatan, untuk baris ketujuh (0), ini berarti bahwa di ruangan tersebut tidak terdapat dokter karena idnya 0 (undefined). Karena tidak ada dokter, sudah pasti tidak ada pasien.
- d. Baris kesembilan berarti banyaknya pasien yang mempunyai obat di inventory-nya.
  - i. Baris kesepuluh (3 1 2 3 4) berarti pasien dengan id 3, mempunyai obat dengan id 1, 2, 3, dan 4 di inventory-nya.
  - ii. Baris kesebelas (5 2 3 4) berarti pasien dengan id 5, mempunyai obat dengan id 2, 3, dan 4 di inventory-nya.

**Jadi INGAT ! Untuk file load dan save pada tugas besar ini, terdapat dua jenis file yang harus dipertimbangkan, yaitu file dengan ekstensi **.txt** dan **.csv****

## File User (user.csv)

Nama Atribut	Deskripsi	Tipe Data
id	ID dari user	INTEGER
username	Username pengguna	STRING
password	Password pengguna	STRING
role	Role pengguna (Manager, Dokter, Pasien)	STRING
riwayat_penyakit	Penyakit yang sedang diidap pasien setelah dilakukan diagnosis oleh dokter.	STRING
suhu_tubuh	Suhu tubuh pasien (Celsius)	FLOAT
tekanan_darah_sistolik	Tekanan darah sistolik (mmHg)	INTEGER
tekanan_darah_diastolik	Tekanan darah diastolik (mmHg)	INTEGER
detak_jantung	Detak jantung (bpm)	INTEGER
saturasi_oksigen	Tingkat saturasi oksigen dalam darah (%)	FLOAT
kadar_gula_darah	Kadar gula darah (mg/dL)	INTEGER
berat_badan	Berat badan pasien (kg)	FLOAT
tinggi_badan	Tinggi badan pasien (cm)	INTEGER
kadar_kolesterol	Kadar kolesterol total (mg/dL)	INTEGER

kadar_kolesterol_ldl	Kadar kolesterol LDL (mg/dL)	INTEGER
trombosit	Jumlah trombosit (ribu/ $\mu$ L)	INTEGER

### File Penyakit (penyakit.csv)

Nama Atribut	Deskripsi	Tipe Data
id	ID penyakit	INTEGER
nama_penyakit	Nama penyakit	STRING
suhu_tubuh_min	Batas minimum suhu tubuh	FLOAT
suhu_tubuh_max	Batas maksimum suhu tubuh	FLOAT
tekanan_darah_sistolik_min	Batas minimum tekanan darah sistolik	INTEGER
tekanan_darah_sistolik_max	Batas maksimum tekanan darah sistolik	INTEGER
tekanan_darah_diastolik_min	Batas minimum tekanan darah diastolik	INTEGER
tekanan_darah_diastolik_max	Batas maksimum tekanan darah diastolik	INTEGER
detak_jantung_min	Batas minimum detak jantung	INTEGER
detak_jantung_max	Batas maksimum detak jantung	INTEGER
saturasi_oksigen_min	Batas minimum saturasi oksigen	FLOAT
saturasi_oksigen_max	Batas maksimum saturasi oksigen	FLOAT
kadar_gula_darah_min	Batas minimum kadar gula darah	INTEGER
kadar_gula_darah_max	Batas maksimum kadar gula darah	INTEGER

berat_badan_min	Batas minimum berat badan (mungkin kurang relevan untuk penyakit, kecuali kondisi tertentu)	FLOAT
berat_badan_max	Batas maksimum berat badan (mungkin kurang relevan untuk penyakit, kecuali kondisi tertentu)	FLOAT
tinggi_badan_min	Batas minimum tinggi badan (mungkin kurang relevan untuk penyakit)	INTEGER
tinggi_badan_max	Batas maksimum tinggi badan (mungkin kurang relevan untuk penyakit)	INTEGER
kadar_kolesterol_min	Batas minimum kadar kolesterol total	INTEGER
kadar_kolesterol_max	Batas maksimum kadar kolesterol total	INTEGER
trombosit_min	Batas minimum jumlah trombosit	INTEGER
trombosit_max	Batas maksimum jumlah trombosit	INTEGER

## File Obat (obat.csv)

File ini berisi data obat yang terdefinisi dalam rumah sakit.

Nama Atribut	Deskripsi	Tipe Data
obat_id	Merepresentasikan id obat	INTEGER
nama_obat	Merepresentasikan nama dari obat tersebut	STRING



## File Obat - Penyakit(obat\_penyakit.csv)

File ini akan relasi data obat dan penyakit yang bisa disembuhkan.

Nama Atribut	Deskripsi	Tipe Data
obat_id	Merepresentasikan id obat	INTEGER
penyakit_id	Merepresentasikan id penyakit yang dapat disembuhkan dengan obat ini	INTEGER
urutan_minum	Merepresentasikan urutan pemberian obat	INTEGER

# Ketentuan dan Batasan

## A. Ketentuan Umum

1. Tugas dikerjakan berkelompok dengan anggota kelompok sebanyak 5–6 orang per kelompok dengan anggota kelompok yang telah ditentukan oleh asisten. Link **pembagian kelompok** serta **pemilihan asisten** dapat dilihat pada tautan berikut:  
[IF1210 Pembagian Kelompok dan Asisten](#)
2. **Template MoM Asistensi** dan **Laporan Tugas Besar** dapat dilihat pada tautan berikut:  
[\[TEMPLATE\] IF1210\\_FormAsistensiTB\\_\[1/2\]\\_K\[XX-Y\].docx](#)  
[\[TEMPLATE\] IF1210\\_LaporanTB\\_K\[XX-Y\].docx](#)
3. Pertanyaan terkait Tugas Besar, dapat diajukan dalam **Spreadsheets QNA** yang ada pada tautan berikut:  
[IF1210\\_QNA\\_Tugas Besar](#)
4. Mekanisme pengerjaan silakan perhatikan bagian [Pengumpulan dan Deliverables](#)

## B. Batasan Pengerjaan

1. Menggunakan bahasa **C**
2. Hanya boleh melakukan include library dasar C, seperti **stdio.h**, **stdlib.h**, dan **string.h**, serta modul-modul serta fungsi yang telah dibuat sendiri. (Apabila ada library lain yang dibutuhkan, bisa dikonfirmasi terlebih dahulu ke asisten)
3. Tidak boleh menggunakan library eksternal dalam bentuk apapun
4. Hanya boleh memakai fungsi-fungsi bawaan dari C yang diajarkan di kelas. Silakan mengimplementasikan sendiri fungsi tambahan yang dibutuhkan.
5. **Tidak boleh membuat temporary CSV sebagai tempat penyimpanan sementara.** Gunakan struktur data yang dibuat sendiri untuk melakukan penyimpanan. **CSV hanya boleh digunakan sebagai penyimpanan ketika dipanggil load dan save.**
6. Program berjalan sebagai satu kesatuan. Main program hanya boleh satu. Main program ini akan memanggil fungsi/prosedur dari modul lain yang berada di file lain.
7. Tugas besar harus diimplementasikan dalam **paradigma prosedural dan fungsional** seperti yang diajarkan di kelas. Tidak boleh menggunakan paradigma lain seperti OOP.
8. Environment pengerjaan dibebaskan. Akan tetapi program harus dapat dikompilasi di **Linux / unix** (untuk pengguna windows, program harus dapat dikompilasi di [WSL](#)).

Kompilasi akan dilakukan dengan menggunakan **makefile**. (Kalau gak bisa dicompile di Linux/Unix system, gak bisa kita nilai 🙏)

### C. Jadwal Pelaksanaan Kegiatan

Kegiatan	Tanggal/Deadline
<b>Launching Tugas Besar</b>	26 Apr 2025
<b>Batas Pengisian Asisten Kelompok</b>	28 Apr 2025, 12.10 WIB
<b>Pengerjaan Tugas Besar:</b>	
• Deadline Asistensi 1	10 Mei 2025
• Deadline Asistensi 2	24 Mei 2025
<b>Pengumpulan Deliverables</b>	
• Milestone 1	11 Mei 2025, 12.10 WIB
• Milestone 2	30 Mei 2025, 12.10 WIB
<b>Demo Tugas Besar</b>	TBA!

### D. Pengerjaan Tugas Besar

1. Asistensi:
  - a. Setiap kelompok akan didampingi oleh seorang asisten pembimbing dari asisten IF1210. Akan disediakan mekanisme dan informasi untuk mengontak asisten pembimbing.
  - b. Setiap kelompok wajib mengontak asisten pembimbing untuk melakukan asistensi **minimum 2 kali** sepanjang pengerjaan Tugas Besar:
    - i. **Asistensi 1:** paling lambat **Sabtu, 10 Mei 2025**
    - ii. **Asistensi 2:** paling lambat **Sabtu, 24 Mei 2025**
  - c. Setiap kelompok wajib melakukan **minimum 2 kali** pengumpulan tugas dengan mekanisme sesuai dengan ketentuan [pengumpulan dan deliverable](#) dengan ketentuan masing-masing milestone adalah sebagai berikut:
    - i. Milestone 1:
      - Paling lambat **Minggu, 11 Mei 2025** pukul **12.10 WIB**
      - Minimum progres  $\pm 40\%$
      - Dengan rekomendasi fitur minimal: **F01, F02, F03, F04, F05, F06, F10, F18**
      - Rekomendasi Laporan minimal: **Template + F00**
    - ii. Milestone 2:
      - Paling lambat **Jumat, 30 Mei 2025** pukul **12.10 WIB**

- d. Hal-hal yang harus diperhatikan:
- Asisten **berhak menolak** jadwal yang diajukan oleh mahasiswa apabila mengontak terlalu dekat dengan deadline asistensi.
  - Maksimal mengontak asisten **paling lambat H-1** dari jadwal asistensi yang diajukan
  - Asistensi dapat dilakukan secara **daring ataupun luring** tergantung dengan ketersediaan asisten, sangat disarankan untuk berdiskusi dengan asisten secara lebih lanjut.
  - Setelah menyepakati jadwal dengan asisten, silakan membuat **Google Calendar** (disertakan link Google Meet / Zoom jika pelaksanaan secara daring) dan mengundang asisten. Format penamaan Google Calendar adalah:  
**[IF1210] Asistensi\_[1/2]\_K[XX-Y]**  
Contoh: *[IF1210] Asistensi\_1\_K99-Z*
- e. Asistensi dapat digunakan untuk melakukan **klarifikasi dan diskusi spesifikasi** tugas besar. Asistensi **tidak bertujuan** untuk meminta *debugging* / mencari Error, karena *obstacle* tersebut adalah wadah kalian dapat belajar memahami pemrograman.
- f. Setiap kali asistensi, praktikan diminta **membuat MoM** (Minutes of Meeting) asistensi dan mengisinya ke dalam form asistensi dengan format yang ditetapkan. MoM asistensi akan dilampirkan ke dalam laporan.

## 2. Demo:

- Setelah masa deadline pengerjaan Tugas Besar, akan dilaksanakan Demo Tugas Besar secara sinkron dengan asisten masing-masing.
- Demo bertujuan untuk mempresentasikan hasil pengerjaan tugas besar.
- Program akan dijalankan di depan asisten dan akan diuji fungsionalitas, validasi, dan juga pemahaman kalian terhadap program.
- Mekanisme demo akan dijelaskan lebih lanjut di kemudian hari.

# Pengumpulan dan Deliverables

1. Untuk tugas ini Anda diwajibkan menggunakan *git version control system* dengan menggunakan sebuah repository **private** di Github Classroom "Labpro-22". Silakan akses Github Classroom pada:

[Github Classroom - Labpro 22](#)

Notes:

- Dalam sehari, hanya bisa kurang lebih 50 orang yang bergabung ke github classroom. Jadi, gantian ya masuknya 🙏
2. Silakan membuat Group dengan format nama "KXX-Y", dengan penjelasan:  
XX: Nomor kelas (ditulis dalam 2 digit, misal: K03)  
Y: Kode kelompok (ditulis dalam 1 huruf, misal: A)  
Contoh: K03-A
  3. Pembuatan Group dilakukan **satu kali** untuk tiap kelompok, setelah dibuat, anggota kelompok lain **tidak perlu membuat Group lagi** dan dapat langsung bergabung dengan Group yang sudah dibuat.
  4. Akan terdapat *repository private* yang dibuat secara **otomatis** untuk tiap Group, silakan gunakan *repository* tersebut untuk pengerjaan Tugas Besar. Repository tersebut akan memiliki nama: **if1210-tubes-2025-kxx-y**
  5. Sumber kode program yang dibuat sesuai standar yang diajarkan di kuliah, yaitu:
    - a. **Menggunakan nama variabel dan file yang berarti.**  
Contoh yang salah: aaa, bbb, akusayangkamu, akucintatugasbesar dsb.
    - b. **Gunakan Case yang konsisten**  
Misal:
      - kebab-case untuk penamaan file
      - camelCase untuk penamaan variable
      - PascalCase untuk penamaan structPanduan case dapat dilihat pada link [berikut](#).
    - c. **Bersih**  
Hanya mengandung bagian-bagian yang diperlukan
    - d. **Well-Commented**
  6. Repository berisi:
    - Folder **src**: Berisi kode program / fungsi-fungsi.
    - Folder **data**: Contoh data yang dapat di load (silakan tambahkan sendiri)
    - Folder **doc**: Berisi laporan dan MoM dalam .pdf

- **README.md**: Deskripsi singkat cara menjalankan program

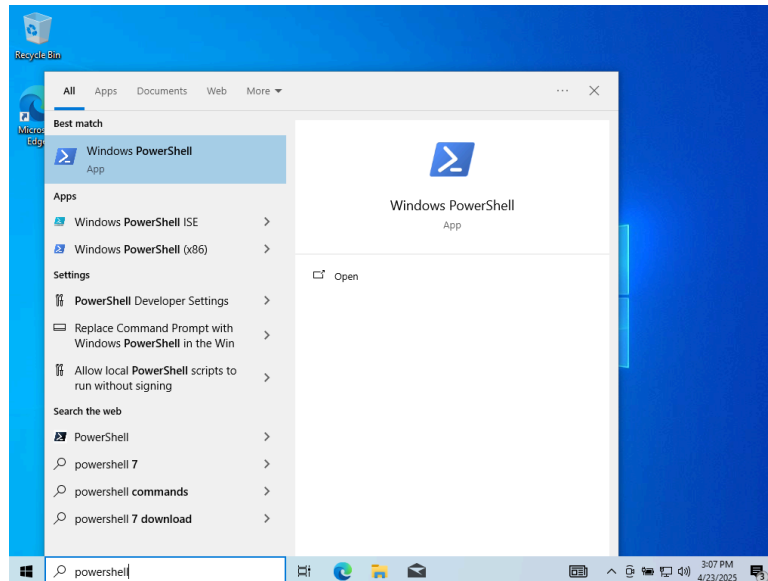
Note: Gunakan template repository yang telah diberikan

7. Laporan dan MoM Tugas Besar dikumpulkan pada GitHub Repository yang telah dibuat pada folder **doc**. Pastikan memiliki format .pdf. Ketentuan pengumpulan dokumen adalah sebagai berikut:
  - a. **Milestone 1**: Dokumen draf laporan sementara dengan format nama **IF1210\_LaporanTB\_1\_K[XX-Y].pdf**
  - b. **Milestone 2**: Dokumen draf laporan sementara dengan format nama **IF1210\_LaporanTB\_2\_K[XX-Y].pdf**
8. Pengumpulan dilakukan dengan membuat **release** dengan major version sesuai dengan deadline milestone terkait, seperti **v1.x** untuk release tag milestone 1. Versi patch dapat digunakan jika ada perbaikan. Contoh: Jika ada revisi pada versi **v1.0**, selama tidak melewati *deadline*, dipersilahkan melakukan revisi kode dan melakukan *release* lagi dengan versi **v1.1**. Penilaian akan dilakukan terhadap versi rilis terakhir.
9. Deliverable yang perlu dikumpulkan disetiap milestone adalah:
  - a. Kode Program
  - b. Laporan
  - c. Data (Jika sudah diperlukan)

# Panduan Instalasi WSL + Makefile

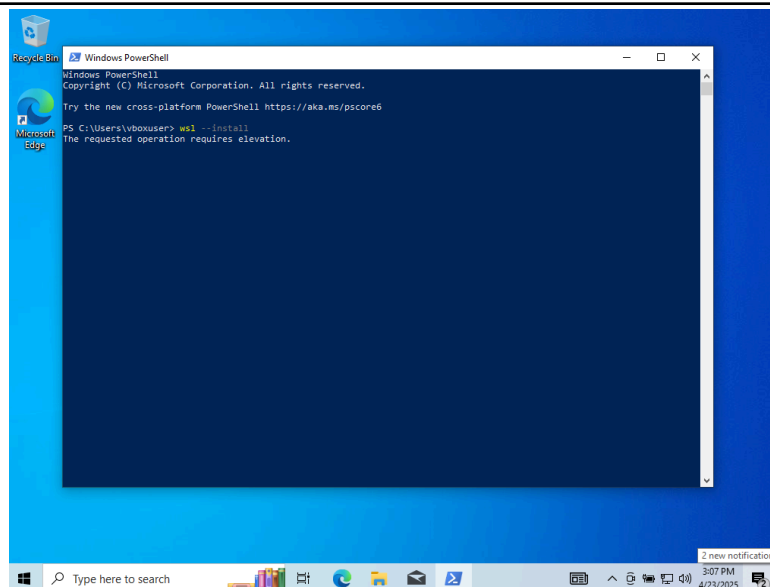
## A. Windows 10/11

1. Buka terminal Powershell

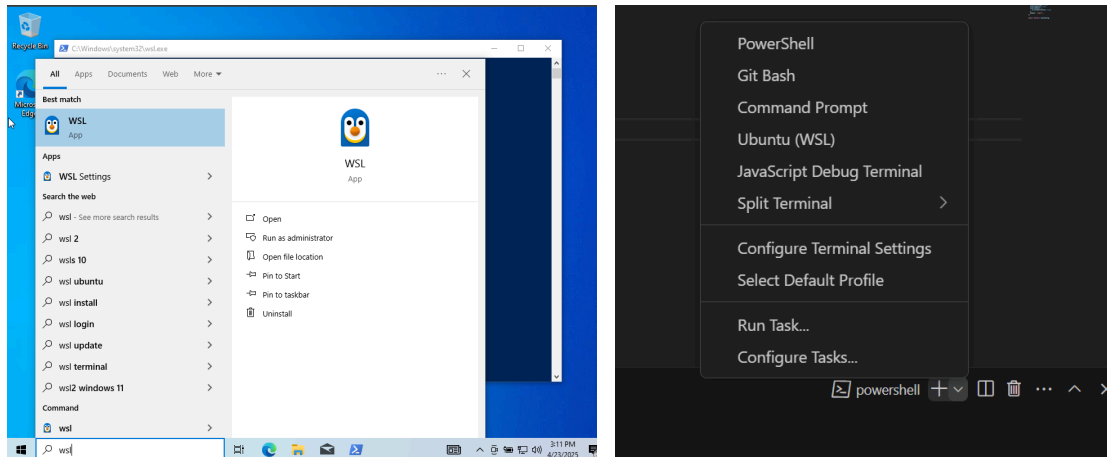


2. run perintah berikut ini untuk menginstall wsl

```
wsl --install
```



3. Buka terminal WSL yang sudah diinstal barusan (Jika menggunakan vscode juga akan ada).



4. Update terlebih dahulu list package yang tersedia

```
sudo apt update
```

5. Install package essential (sudah termasuk makefile dan c)

```
sudo apt install build-essential
```

6. Makefile dan compiler gcc sudah terinstall. Coba dengan run perintah berikut

```
make --version  
gcc --version
```

7. Referensi tambahan: [WSL Installation](#) (Recommended to upgrade your WSL 1 to WSL 2)

## B. Mac

1. Tidak perlu ada setup tambahan
2. Pastikan `make` dan `gcc` sudah terinstall dengan menggunakan command berikut

```
make --version  
gcc --version
```



# Referensi

- [Wikipedia - Linear Congruential Generator \(LCG\)](#)
- [Wikipedia - Run Length Encoding \(RLE\)](#)
- [Devzery - Lexicographic Order](#)
- [Case-Sensitive and Case-Insensitive String](#)
- [GeeksforGeeks - Arguments in C](#)
- [Git Best Practices](#)
- [Git Semantic Commit Message](#)
- [Makefile - Reference 1](#)
- [Makefile - Reference 2](#)
- [Makefile + GCC - Reference 3 \(recommended after having basic makefile understanding\)](#)
- [Microsoft - WSL Installation](#)

## Extras



"Bang udah bang 🙏"  
- valen -

"Tub Tub Tub Tub Tub TUBES"  
- dhafin -



"Selamat farming bro. Semoga dapat thingamabob"  
- Chai -

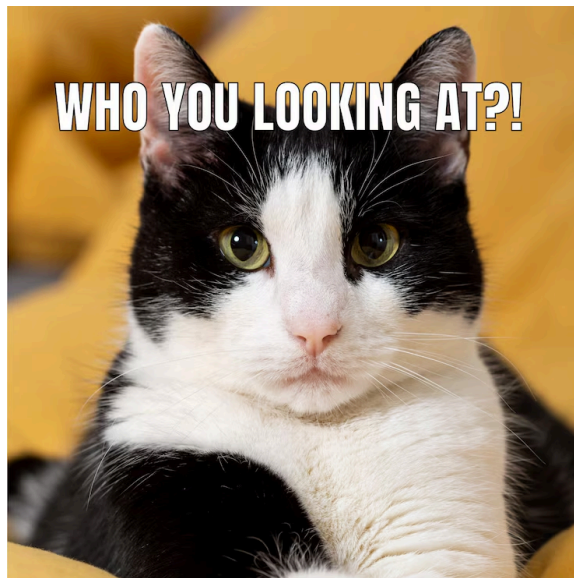


"Meow"

- Adril -

"Smngt"

- Rayhan -



"Smangat brok"

- Albert-

"Selamat Datang Anak Muda"

Iki

"Janlup tidur"

- Roihan -

"Selamat menyusuri cafe - cafe di Jatinangor"

- Maulvi -



"Beneran cooked ini bang 🤔🤔"

- Jendra -

my\_heart\* (\* you)(flowers[]);

"What are 'you'?"

- Nicho -