Spesifikasi Tugas Besar
IF1210 Dasar Pemrograman 2022

Tim Laboratorium Programming 2019

Versi : **9**
Tgl. Revisi Terakhir : **19 April 2022**
Deadline : **25 April 2022, 07:00**

Revisi 1: 29 Maret 2022
Revisi 2: 31 Maret 2022
Revisi 3: 2 April 2022
Revisi 4: 4 April 2022
Revisi 5: 9 April 2022 → Update Daftar Isi
Revisi 6: 11 April 2022
Revisi 7: 13 April 2022
Revisi 8: 19 April 2022

Daftar Isi

Deskripsi Persoalan

"BNMO rusak setelah dibanting oleh Indra yang stress akibat gacha yang rugi terus dan kuliah di ITB"

BNMO (dibaca: Binomo) adalah sebuah robot *game* milik Indra dan Doni yang membantu mereka melepas stress ketika mendapatkan tugas selama di Institut Teknologi Bandung. BNMO dulunya memiliki sistem inventarisasi & toko game yang baik. Indra dan Doni menjalani kuliah 2 semester di ITB dan merasa kesulitan dan stress. Doni menghabiskan waktu dengan BNMO untuk bermain *game*. Namun, Indra lebih suka bersenang-senang bermain *gacha*, akan tetapi ia rugi terus. Sehingga, pada suatu saat, Indra membanting BNMO sehingga BNMO pun rusak. Doni merasa depresi saat ia tahu BNMO rusak. Doni pun segera memperbaiki BNMO dan ia pun meminta bantuan kalian untuk memperbaiki BNMO karena Doni tidak cukup ahli dalam ngoding.

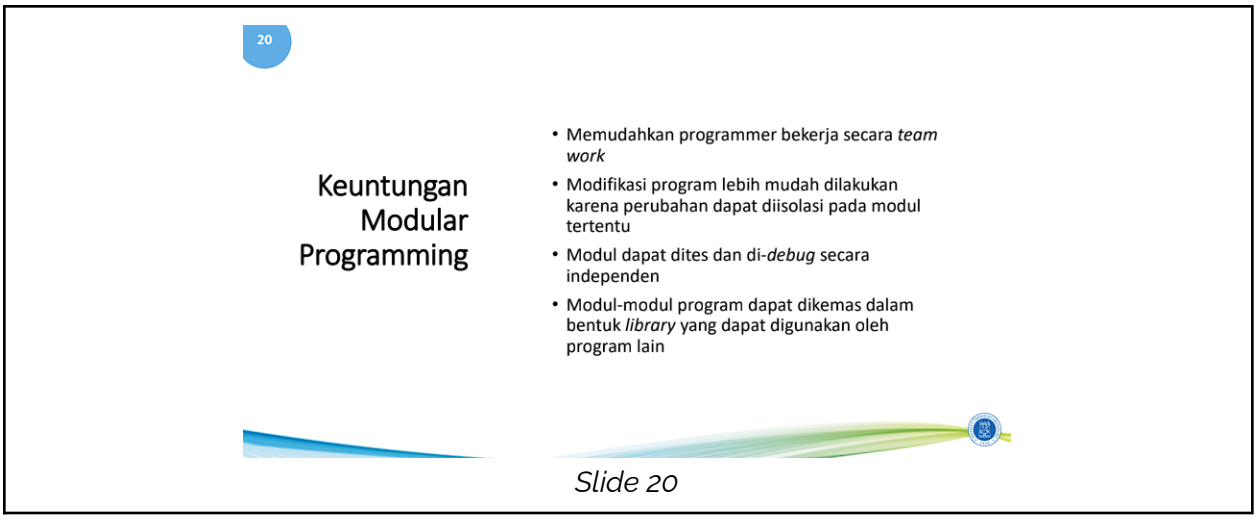
Spesifikasi Program

Terdapat kebutuhan fungsional wajib yang diperlukan oleh BNMO, 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 pada alur jalannya program, melainkan pada validasi yang dilakukan.

F01 - Dekomposisi, Abstraksi dan Generalisasi Pola dengan Modular Programming

Modular Programming merupakan suatu teknik pemrograman untuk memecah/dekomposisi suatu program yang besar menjadi beberapa bagian program yang lebih kecil. Jika gambar pada bawah ini terlalu kecil untuk dilihat, bisa melihat langsung pada Slide Kuliah "Dekomposisi, Abstraksi, Generalisasi Pola dalam Konteks Pemrograman Prosedural."

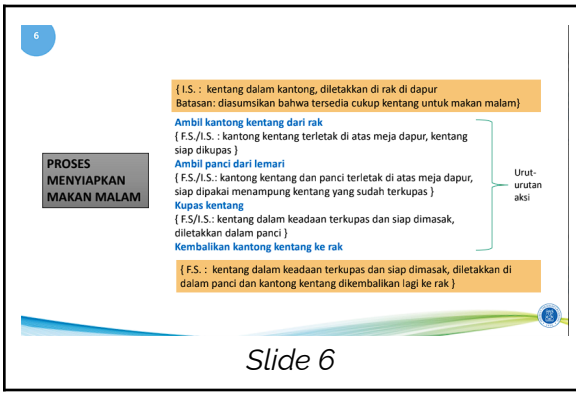
Khusus untuk "F01 - Dekomposisi, Abstraksi dan Generalisasi Pola", harus dilakukan oleh semua anggota.



Slide 20

Hal ini bisa dilakukan dengan cara-cara sebagai berikut:

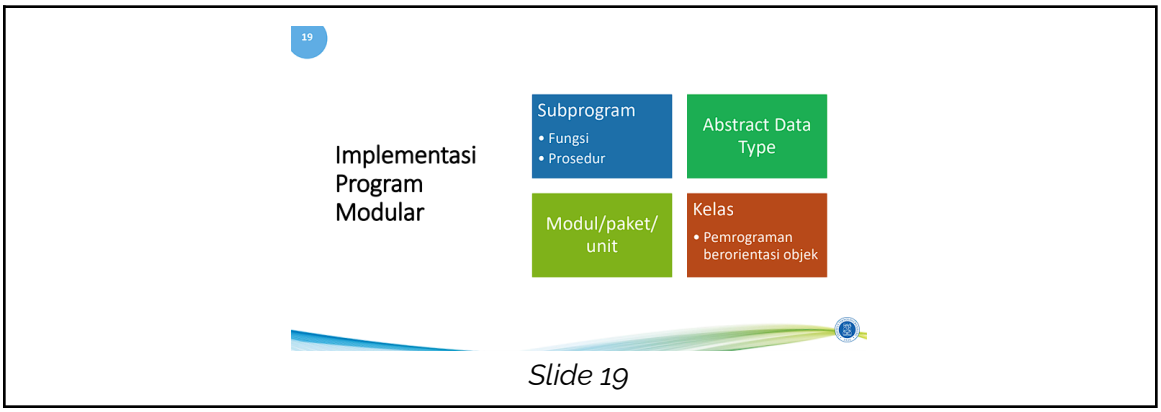
1. Memecah bagian program yang besar ke beberapa fungsi. Contoh: Fungsional Register terdiri atas pemanggilan fungsi validasi username, penambahan data, dan prosedur pengiriman pesan umpan balik.



Slide 6

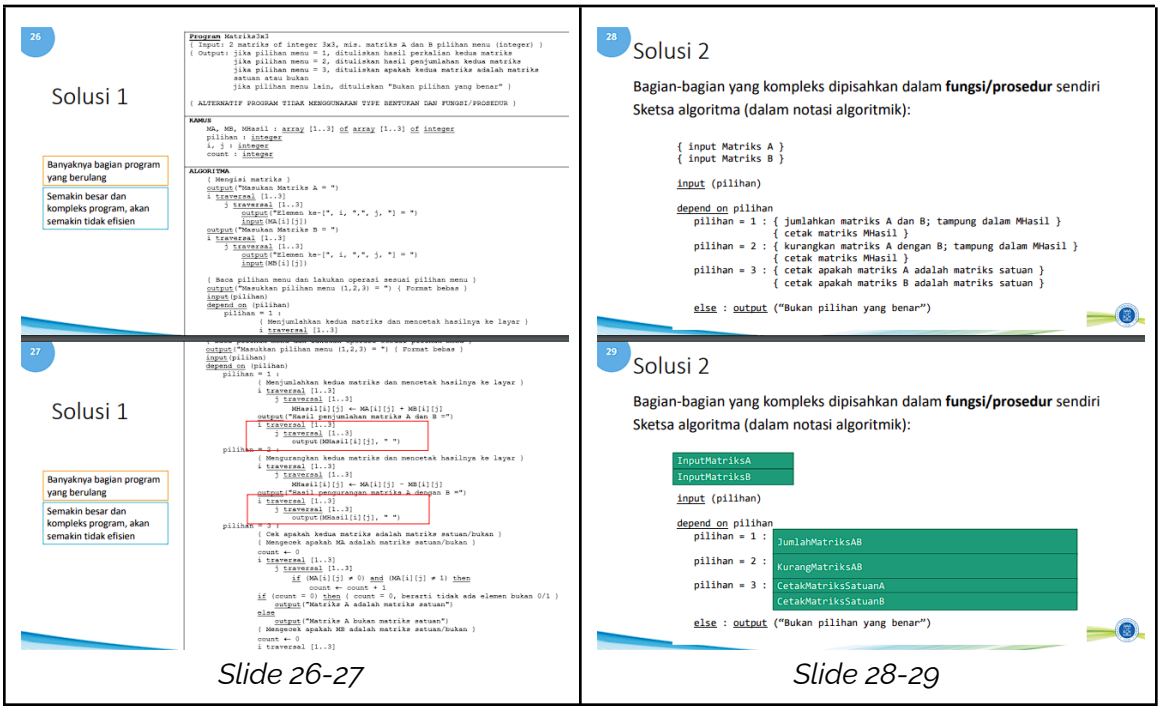
Slide 7

2. Memecah fungsi-fungsi pada file terpisah dengan paket/modul. Referensi yang dapat digunakan adalah sebagai berikut, Referensi 1, Referensi 2 dan Referensi 3.



Slide 19

3. Merancang dari awal fungsi apa saja yang diperlukan sehingga tidak perlu mengimplementasikan berulang-ulang. Contohnya, banyak dilakukan pengecekan role untuk user yang sudah login, maka rancanglah suatu fungsi untuk melakukan validasi role tersebut sehingga hanya perlu memanggil fungsi tersebut setiap kali membutuhkan.



Slide 26-27 *Slide 28-29*

4. Memanggil fungsi yang sudah ada apabila memerlukan skema yang sama. Contoh: Login membutuhkan validasi username, dan sudah ada fungsi validasi username, maka langsung gunakan itu, tidak perlu implementasi dua kali. Apabila suatu skema

diperlukan berkali-kali namun belum ada fungsinya, lebih baik dijadikan sebuah fungsi daripada melakukan *copy paste* skema di mana-mana.

F02 - Register

Akses: Admin

Admin dapat mendaftarkan pengguna baru dengan memasukkan nama, username, dan password. Pengguna yang mendaftar otomatis memiliki role "user". Pastikan username bersifat unik. Fungsi ini tidak dapat membuat user dengan role admin; untuk membuat user admin dapat mengedit file penyimpanan. Username hanya dapat mengandung alfabet A-Za-z, *underscore* "_", *strip* "-", dan angka 0-9.

```
# Sebagai admin… dan misalkan username belum terdaftar di sistem.
>>> register
Masukan nama: Fabian Savero
Masukan username: fabian_si_banyak_fansnya00
Masukan password: yukmainbinomo

Username fabian_si_banyak_fansnya00 telah berhasil register ke dalam "Binomo".
>>>
```

```
# Sebagai admin… dan misalkan username sudah terdaftar pada sistem.
>>> register
Masukan nama: Fabian Savero
Masukan username: fabian_si_banyak_fansnya00
Masukan password: yukmainbinomo

Username fabian_si_banyak_fansnya00 sudah terpakai, silakan menggunakan username lain.
>>>
```

F03 - Login

Akses: User dan Admin

Saat masuk ke dalam aplikasi, pengguna bisa login dengan memasukkan username dan password. Bila username dan password yang diinput cocok dengan username dan password pada file user, maka pengguna berhasil login. Untuk alur login dibebaskan; tidak harus mengetik "login" dulu diperbolehkan (misal, ketika sistem menyala user diwajibkan login, namun pastikan kalau belum login tidak bisa melakukan apa-apa selain login).

```
>>> login
Masukan username: fabian_si_banyak_fansnya00
Masukan password: yukmainbinomo
# Menuliskan greeting nama.
Halo Fabian Savero! Selamat datang di "Binomo".
# Meminta perintah berikutnya, bisa menjalankan perintah lain sesuai dengan akses yang dimiliki.
>>>
```

```
# Misalkan dilakukan login dengan username yang tidak ditemukan/salah atau password yang salah.
>>> login
Masukan username: fabian_si_banyak_fansnya00
Masukan password: gakmaumainbinomo

Password atau username salah atau tidak ditemukan.
# Meminta perintah berikutnya, hanya bisa melakukan login.
>>>
```

```
# Misalkan belum login dan “mencoba_sesuatu” adalah perintah yang valid.
>>> mencoba_sesuatu
Maaf, anda harus login terlebih dahulu untuk mengirim perintah selain “login”
>>>
```

```
# Misalkan terdapat perintah yang valid berupa “hanya_user” yang hanya dapat dilakukan oleh user. Namun, yang login adalah seorang admin.
>>> hanya_user
Maaf, anda harus menjadi user untuk melakukan hal tersebut.
>>>
```

```
# Misalkan terdapat perintah yang valid berupa “hanya_admin” yang hanya dapat dilakukan oleh admin. Namun, yang login adalah seorang user.
>>> hanya_admin
Maaf, anda tidak memiliki izin untuk menjalankan perintah berikut. Mintalah ke administrator untuk melakukan hal tersebut.
>>>
```

F04 - Menambah Game ke Toko Game

Akses: Admin

Penambahan game pada toko game dilakukan melalui pengisian informasi game yang akan ditambahkan. Program lalu akan melakukan validasi apakah semua informasi yang dibutuhkan telah di*input* oleh admin. Apabila terdapat informasi yang belum dimasukkan oleh pengguna, program akan meminta *input* lagi kepada pengguna. Hal ini akan dilakukan terus menerus sampai pengguna telah melakukan *input* semua informasi game. ~~Game baru yang telah ditambahkan akan disimpan pada file csv.~~

```
>>> tambah_game
Masukkan nama game:
Masukkan kategori: Adventure
Masukkan tahun rilis: 2022
Masukkan harga: 100.000
Masukkan stok awal: 1

# Mengembalikan pesan error
Mohon masukkan semua informasi mengenai game agar dapat disimpan BNMO.
Masukkan nama game:
Masukkan kategori:
Masukkan tahun rilis:
Masukkan harga:
Masukkan stok awal:
```

```
>>> tambah_game
Masukkan nama game: BNMO - Play Along With Crypto
Masukkan kategori: Adventure
Masukkan tahun rilis: 2022
Masukkan harga: 100.000
Masukkan stok awal: 1

# Mengembalikan pesan sukses
Selamat! Berhasil menambahkan game BNMO - Play Along With Crypto.
# Data yang ditambahkan hanya data pada memori/variabel. Data pada storage/CSV bertambah hanya jika dilakukan perintah save, jika tidak dilakukan, penambahan tidak disimpan pada file.
>>>
```

F05 - Mengubah Game pada Toko Game

Akses: Admin

Mengubah game pada toko game dilakukan melalui pengisian informasi game yang akan diubah. Setelah itu game dengan informasi baru yang telah diubah akan disimpan pada file csv. Admin juga tidak harus mengisi semua field selain field ID, sehingga saat admin ingin membiarkan value field tertentu agar sesuai dengan value field data yang lama, admin dapat melewatinya dengan mengosongkan. **Tidak bisa mengubah stock** pada fungsional ini. Field ID tidak dapat diubah, hanya digunakan sebagai *keyword* pencari game.

```
# Misal terdapat data sebagai berikut:
[nama, kategori, tahun_rilis, harga, stock]
# Lalu data sebelum diubah adalah sebagai berikut
[Punten, Adventure, 2022, 100.000, 1]
>>> ubah_game
Masukkan ID game: G001
Masukkan nama game:
Masukkan kategori:
Masukkan tahun rilis: 2023
Masukkan harga: 120.000
# Data akan berubah menjadi yang seperti dibawah ini
[Punten, Adventure, 2023, 120.000, 1]
# Data yang diupdate hanya data pada memori/variabel. Data pada storage/CSV berubah hanya jika dilakukan perintah save, jika tidak dilakukan, pengubahan tidak disimpan pada file.
>>>
```

```
# Pada kasus normal, jika semua field data seperti pada input berikut, maka semua field pada data game yang diubah otomatis ikut berubah
>>> ubah_game
Masukkan ID game: G001
Masukkan nama game: BNMO - Play Along With Crypto
Masukkan kategori: Adventure
Masukkan tahun rilis: 2022
Masukkan harga: 100.000
>>>
```

F06 - Mengubah Stok Game di Toko

Akses: Admin

Mengubah stok sebuah game pada toko dilakukan melalui input ID dan besar perubahan stok yang ingin dilakukan. Saat dilakukan pengubahan stok suatu game, perlu dilakukan validasi untuk memastikan stok game tersebut tetap valid setelah pengubahan (tidak negatif). Bila stok suatu game bernilai nol setelah pengubahan, tidak perlu dihapus dari sistem.

```
# Misal terdapat data sebagai berikut:
[ID, nama, kategori, tahun_rilis, harga, stock]
# Lalu data sebelum diubah adalah sebagai berikut
[G001, Punten, Adventure, 2022, 100.000, 1]
>>> ubah_stok
Masukkan ID game: G001
Masukkan jumlah: 10
Stok game Punten berhasil ditambahkan. Stok sekarang: 11
# Data yang diupdate hanya data pada memori/variabel. Data pada storage/CSV berubah hanya jika dilakukan perintah save, jika tidak dilakukan, pengubahan tidak disimpan pada file.
>>>
```

```
# Misal terdapat data sebagai berikut:
[ID, nama, kategori, tahun_rilis, harga, stock]
# Lalu data sebelum diubah adalah sebagai berikut
[G001, Punten, Adventure, 2022, 100.000, 1]
>>> ubah_stok
Masukkan ID game: G001
Masukkan jumlah: -1
Stok game Punten berhasil dikurangi. Stok sekarang: 0
>>>
```

```
# Misal terdapat data sebagai berikut:
[ID, nama, kategori, tahun_rilis, harga, stock]
# Lalu data sebelum diubah adalah sebagai berikut
[G001, Punten, Adventure, 2022, 100.000, 1]
>>> ubah_stok
Masukkan ID game: G001
Masukkan jumlah: -10
Stok game Punten gagal dikurangi karena stok kurang. Stok sekarang: 1 (< 10)
>>>
```

```
# Misal terdapat data sebagai berikut:
[ID, nama, kategori, tahun_rilis, harga, stock]
# Lalu data sebelum diubah adalah sebagai berikut
[G001, Punten, Adventure, 2022, 100.000, 1]
>>> ubah_stok
Masukkan ID game: G010

Tidak ada game dengan ID tersebut!
>>>
```

F07 - Listing Game di Toko Berdasarkan ID, Tahun Rilis dan Harga

Akses: User dan Admin

Agar pengguna dapat melihat *game-game* yang ada di toko, toko harus bisa memberikan daftar *game* yang dimiliki. Beberapa pengguna mungkin merupakan *gamer* yang *up-to-date* terhadap perkembangan game, sehingga ingin melihat *game* yang terbaru, sedangkan beberapa di antaranya merupakan *gamer* yang suka mengoleksi *game* klasik. Selain itu juga ada pengguna yang merupakan *gamer* dengan budget pas-pasan, sehingga perlu berpikir dua kali ketika membeli *Fifa 22*.

Oleh karena itu, selain menyediakan daftar *game*, toko juga harus bisa melakukan *sorting* terhadap *game* berdasarkan harga dan tahun rilisnya. Pengguna bisa memberikan inputan berupa skema *sorting* dari daftar *game* yang ingin dilihat ke dalam aplikasi. Terdapat 2 skema sorting, yaitu berdasarkan tahun rilis **atau** harga (**satu saja, tidak ada kasus** di sort berdasarkan **dua attribut**), urutan bisa *ascending* atau *descending*. Pastikan skema sorting tervalidasi. Parsing input skema sorting dibebaskan. Jika skema sorting dikosongkan, akan di sort berdasarkan ID *ascending*.

```
>>> list_game_toko
Skema sorting : tahun-
# Menuliskan daftar game di toko dengan sorting tahun secara descending (menurun), ditandai dengan tanda “-” (minus)
# Pada contoh ini formatnya adalah ID | NAMA | Harga | Kategori | Tahun rilis | Stok
1. GAME009 | Elder Ring    | 861000  | Adventure | 2022 | 1
2. GAME010 | GoW: Ragnarok | 1000000 | Adventure | 2022 | 5
3. GAME008 | AC Unity      | 430000  | Adventure | 2014 | 0
>>>
# Data yang diperlihatkan data pada memori/variabel
```

```
>>> list_game_toko
Skema sorting: harga+
# Menuliskan daftar game di toko dengan sorting harga secara ascending (menaik), ditandai dengan tanda “+” (plus)
1. GAME008 | AC Unity      | 430000  | Adventure | 2014 | 0
2. GAME009 | Elder Ring    | 861000  | Adventure | 2022 | 1
3. GAME010 | GoW: Ragnarok | 1000000 | Adventure | 2022 | 5
>>>
```

```
>>> list_game_toko
Skema sorting:
# Menuliskan daftar game di toko dengan tanpa sorting / berdasarkan id ascending (menaik)
1. GAME008 | AC Unity      | 430000 | Adventure  | 2014 | 0
2. GAME009 | Elder Ring    | 861000 | Adventure  | 2022 | 1
3. GAME010 | GoW: Ragnarok | 1000000 | Adventure | 2022 | 5
>>>
```

```
>>> list_game_toko
Skema sorting: waifu++
# Menuliskan pesan error karena skema tidak valid
Skema sorting tidak valid!
>>>
```

F08 - Membeli Game

Akses: User

User dapat membeli Game dengan menggunakan prosedur ini. Game yang telah dibeli akan masuk ke list Game yang dimiliki User. Game hanya dapat dibeli user yang sama sebanyak satu kali. Terdapat 1 parameter yang wajib diisi pada prosedur ini, yaitu ID Game yang akan dibeli user.

```
# Apabila game belum dimiliki, user memiliki saldo yang cukup, dan terdapat stok pada toko
>>> buy_game
Masukkan ID Game: GAME404

Game “HTTP” berhasil dibeli!
>>>
# Data yang diubah hanya data pada memori/variabel. Data pada storage/CSV berubah hanya jika dilakukan perintah save, jika tidak dilakukan, perubahan tidak disimpan pada file.
```

```
# Apabila game sudah dimiliki user
>>> buy_game
Masukkan ID Game: GAME404

Anda sudah memiliki Game tersebut!
>>>
```

```
# Apabila user tidak memiliki saldo yang cukup
>>> buy_game
Masukkan ID Game: GAME404

Saldo anda tidak cukup untuk membeli Game tersebut!
>>>
```

```
# Apabila tidak terdapat stok pada toko

>>> buy_game
Masukkan ID Game: GAME404

Stok Game tersebut sedang habis!
>>>
```

F09 - Melihat Game yang dimiliki

Akses: User

Prosedur ini memberikan daftar game yang dimiliki pengguna. Tidak ada aturan khusus untuk urutan *game* yang ditampilkan. Tampilkan pesan khusus ketika user tidak memiliki *game*.

```
>>> list_game
# Menuliskan pesan error karena user tidak memiliki game.
Maaf, kamu belum membeli game. Ketik perintah beli_game untuk beli.
>>>
```

```
>>> list_game
# Menuliskan pesan error karena user tidak memiliki game.
# Urutan kolom:
# ID | Nama game | Kategori | Tahun Rilis | Harga
Daftar game:
1. GAME001 | BNMO - Play Along With Crypto | Adventure   | 2022 | 100.000
2. GAME069 | Python Gemink                 | Programming | 1991 | 69.000
3. GAME666 | Hehehe                        | Comedy      | 2012 | 666.000
>>>
```

F10 - Mencari Game yang dimiliki dari ID dan tahun rilis

Akses: User

Prosedur ini digunakan untuk mendapatkan informasi game sesuai dengan query yang diminta oleh pengguna pada inventory. Terdapat 2 parameter yang dapat digunakan, yaitu ID Game dan Tahun Rilis Game. Parameter bersifat **tidak wajib** diisi. Jadi, apabila user ingin mencari game dengan kategori tertentu saja, Ia dapat mengosongkan parameter lain dengan tidak mengisi input apapun. Ketika semua parameter dikosongkan, maka sistem akan menampilkan semua list game yang dimiliki oleh user tersebut.

```
# Apabila game dimiliki dan user memasukkan semua parameter

>>> search_my_game
Masukkan ID Game: GAME404
Masukkan Tahun Rilis Game: 2022

Daftar game pada inventory yang memenuhi kriteria:
1. GAME404 | Game Error Not Found | 123.000 | Unknown | 2022
>>>
```

```
# Apabila game tidak dimiliki dan user memasukkan semua parameter

>>> search_my_game
Masukkan ID Game: GAME123
Masukkan Tahun Rilis Game: 2022

Daftar game pada inventory yang memenuhi kriteria:
Tidak ada game pada inventory-mu yang memenuhi kriteria
>>>
```

```
# Apabila game dimiliki dan user tidak memasukkan beberapa/semua parameter

>>> search_my_game
Masukkan ID Game:
Masukkan Tahun Rilis Game: 2022

Daftar game pada inventory yang memenuhi kriteria:
1. GAME404 | Game Error Not Found | 123.000 | Unknown | 2022
2. GAME555 | Hahahaha             | 55.000  | Comedy  | 2022
>>>
```

```
# Apabila game tidak dimiliki pada toko dan user tidak memasukkan semua/beberapa parameter
```

```
>>> search_my_game
Masukkan ID Game:
Masukkan Tahun Rilis Game: 2011

Daftar game pada inventory yang memenuhi kriteria:
Tidak ada game pada inventory-mu yang memenuhi kriteria
>>>
```

F11 - Mencari Game di Toko dari ID, Nama Game, Harga, Kategori dan Tahun Rilis

Akses: Admin dan User

Seperti fungsional sebelumnya (F09), namun kali ini pencarian dilakukan pada toko. Terdapat 5 parameter yang dapat digunakan, yaitu ID Game, Nama Game, Harga, Kategori dan Tahun Rilis Game. Parameter bersifat **tidak wajib** diisi.

```
# Apabila game terdapat pada toko dan user memasukkan semua parameter

>>> search_game_at_store
Masukkan ID Game: GAME001
Masukkan Nama Game: BNMO - Play Along With Crypto
Masukkan Harga Game: 100.000
Masukkan Kategori Game: Adventure
Masukkan Tahun Rilis Game: 2022

Daftar game pada toko yang memenuhi kriteria:
# ID | NAMA  | Harga | Kategori | Tahun rilis | stok
1. GAME001 | BNMO - Play Along With Crypto | 100.000 | Adventure | 2022 | 1
>>>
```

```
Contoh sama seperti F10 hanya saja memiliki 5 Parameter dan mencarinya di Toko.
```

F12 - Top Up Saldo

Akses: Admin

Prosedur ini adalah sebuah prosedur untuk menambahkan saldo kepada User. Masukan menerima sebuah username dan sejumlah saldo. Saldo ini kemudian ditambahkan ke User dengan username yang sesuai. Username perlu dilakukan validasi. Saldo dapat bernilai minus untuk mengurangi. Namun, apabila masukan saldo negatif, perlu dilakukan validasi.

```
# Sebagai admin…

>>> topup
Masukan username: indra_krenz
Masukan saldo: 10000

Top up berhasil. Saldo Indra Krenz bertambah menjadi 15000.
```

```
# misal saldo indra_krenz awalnya adalah 5000, dan di-top up sebanyak 10000. Sehingga
saldonya menjadi 15000. Saldo bisa bernilai minus untuk mengurangi.
# Data yang diubah hanya data pada memori/variabel. Data pada storage/CSV berubah hanya
jika dilakukan perintah save, jika tidak dilakukan, perubahan tidak disimpan pada file.
>>>
```

```
# Sebagai admin… namun saldo gagal validasi

>>> topup
Masukan username: indra_krenz
Masukan saldo: -10000

Masukan tidak valid.

# misal saldo indra_krenz awalnya adalah 5000, dan akan dikurangi 10000. Tetapi karena
10000 > 5000, maka proses pengurangan saldo dibatalkan.
>>>
```

```
# Sebagai admin… dan misalkan “doraemonangis” tidak terdaftar pada sistem.

>>> topup
Masukan username: doraemonangis
Masukan saldo: 10000

Username “doraemonangis” tidak ditemukan.

# misal saldo tidak bertambah karena username tidak ditemukan.
>>>
```

F13 - Melihat Riwayat Pembelian

Akses: User

Setelah melakukan banyak pembelian *game*, pengguna bisa saja ingin melihat riwayat pembelian *game* karena ingin mengetahui berapa harga beli dan/atau kapan pengguna membeli *game-game* yang sudah dimiliki. Tampilkan pesan khusus ketika pengguna tidak memiliki riwayat pembelian *game*.

```
>>> riwayat
# Menuliskan pesan error karena user tidak memiliki riwayat pembelian game.
Maaf, kamu tidak ada riwayat pembelian game. Ketik perintah beli_game untuk membeli.
>>>
```

```
>>> riwayat
# Menuliskan pesan error karena user tidak memiliki game.
# Urutan kolom:
# ID | Nama game | Harga | Tahun Beli |
Daftar game:
1. GAME001 | BNMO - Play Along With Crypto | 100.000 | 2022 |
2. GAME069 | Python Gemink                 | 100.000 | 2001 |
3. GAME666 | Hehehe                        | 100.000 | 2021 |
>>>
```

F14 - Help

Akses: Admin dan User

Seperti namanya, prosedur ini adalah prosedur untuk memberikan panduan penggunaan sistem. Format keluaran dibebaskan, yang penting keluaran memberikan arahan yang cukup jelas dengan akses yang bersesuaian. Tidak perlu melakukan login dulu untuk menggunakan menu ini.

```
# Sebagai admin…

>>> help
============ HELP ============
1. register - Untuk melakukan registrasi user baru
2. login - Untuk melakukan login ke dalam sistem
3. tambah_game - Untuk menambah game yang dijual pada toko
4. list_game_toko - Untuk melihat list game yang dijual pada toko
# ...dan seterusnya untuk fungsi yang hanya bisa diakses admin
>>>
```

```
# Sebagai user…

>>> help
============ HELP ============
1. login - Untuk melakukan login ke dalam sistem
2. list_game_toko - Untuk melihat list game yang dijual pada toko
# ...dan seterusnya untuk fungsi yang hanya bisa diakses user
>>>
```

F15 - Load

Prosedur ini digunakan untuk melakukan loading data ke dalam sistem. Prosedur ini akan otomatis dijalankan ketika sistem mulai pertama kali bila diberikan input nama folder yang berisi file penyimpanan. Semua file penyimpanan dalam suatu folder dijamin ada dan memiliki nama yang fixed, seperti yang tertera pada bagian struktur data eksternal. Namun untuk folder, harus dilakukan validasi.

Masukan nama direktori ketika hendak menjalankan script lewat argumen. Cara membaca argumen dari Python bisa dibaca lebih lanjut menggunakan `argparse` pada link berikut.

```
~$ python program_binomo.py nama_folder

Loading…
# Panggil prosedur load data
Selamat datang di antarmuka “Binomo”
# Meminta perintah berikutnya… (cth: register, login, dll.)
>>>
```

```
~$ python program_binomo.py

Tidak ada nama folder yang diberikan!
```

```
Usage: python program_binomo.py <nama_folder>
# Program selesai
```

```
# Misalkan folder123 tidak ada
~$ python program_binomo.py folder123

Folder “folder123” tidak ditemukan.
# Program selesai
```

F16 - Save

Akses: Admin, User

Prosedur ini digunakan untuk melakukan penyimpanan data ke dalam file setelah dilakukan perubahan. Misalnya, setelah melakukan pembelian atau penjualan tanpa di-*save*, data tersebut tidak akan tersimpan pada file. Beberapa aturan terkait save data:

1. Bila nama folder belum ada, buat folder dan simpan file csv ke dalam folder tersebut. Sebaliknya, bila nama folder sudah ada, tidak perlu menghapus folder yang ada.
2. Bila nama file sudah ada, hapus file tersebut dan ganti dengan yang baru (*replace/overwrite*). Sebaliknya, buat file baru dan simpan ke file tersebut.

Untuk membaca isi folder dapat menggunakan fungsi `os.walk` dengan melakukan `import os` terlebih dahulu. Lebih lengkapnya dapat dibaca pada link berikut. Implementasi lainnya diperbolehkan selama masih terdapat dalam batasan yang diberikan.

```
>>> save
Masukkan nama folder penyimpanan: 2021-03-19_235959

Saving...
Data telah disimpan pada folder 2021-03-19_235959!
>>>
```

F17 - Exit

Seperti namanya, fungsi ini adalah fungsi untuk keluar dari aplikasi. Fungsi dapat menerima huruf kecil maupun besar. Pastikan masukan valid. Kalau tidak valid, bisa tanyakan kembali pertanyaannya.

```
>>> exit
# Misalkan, input tidak valid
Apakah Anda mau melakukan penyimpanan file yang sudah diubah? (y/n) a
# Maka program menanyakan ulang. Input harus berlaku untuk Y atau N dalam huruf kecil maupun besar.
Apakah Anda mau melakukan penyimpanan file yang sudah diubah? (y/n) Y
# Menjalankan prosedur save dan program selesai
```

```
>>> exit
```

```
Apakah Anda mau melakukan penyimpanan file yang sudah diubah? (y/n) n
# Program selesai dan tidak menyimpan data pada file
```

Struktur Data File Eksternal

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

Karena Python di komputer Mobita corrupt, komputer tersebut tidak bisa menggunakan `.split()`, sehingga kalian ***HARUS*** membuat CSV parser sendiri!

Berikut adalah contoh isi file csv-nya untuk file game:

```
id;nama;kategori;tahun_rilis;harga;stok
GAME001;BNMO - Play Along With Crypto;Adventure;2022;100000;1
GAME002;Dasar Pemrograman;Coding;2022;0;10
GAME069;Python Gemink;Coding;1991;69000;999
```

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*.

File User (user.csv)

`id`	User ID, format dibebaskan. Dibuat secara otomatis. Disarankan increment integer, yaitu 1, 2, 3, dst.
`username`	Username untuk masing-masing pengguna, bersifat unik; silakan verifikasi dalam program.
`nama`	Nama Lengkap dari pengguna yang terdaftar.
`password`	Password dari masing-masing pengguna, tidak perlu dilakukan *hashing*. Tulis dalam bentuk *ciphered* jika mengerjakan bonus.
`role`	Role dari pengguna (Admin dan User)
`saldo`	*self-explained.*

File Game (game.csv)

CSV untuk menyimpan game yang tersedia pada toko.

`id`	ID Game, format `GAME<angka>`, contoh GAME001, GAME123
`nama`	Nama game.
`kategori`	Kategori game.
`tahun_rilis`	Tahun game dirilis.
`harga`	Harga game.

`stok`	Stok game.

File Riwayat (riwayat.csv)

`game_id`	ID Game, format `GAME<angka>`, contoh GAME001, GAME123
`nama`	Nama game.
`harga`	Harga game.
`user_id`	User ID.
`tahun_beli`	Tahun pembelian game.

File Kepemilikan (kepemilikan.csv)

`game_id`	ID Game, format `GAME<angka>`, contoh GAME001, GAME123
`user_id`	User ID yang memiliki game tersebut

Spesifikasi Bonus

Bonus bisa dikerjakan beberapa, semua atau tidak sama sekali. Nilai maksimal bisa dicapai dengan realisasi fungsionalitas utama, sehingga prioritaskan fungsionalitas utama terlebih dahulu sebelum menjalankan spesifikasi bonus. Terdapat tiga spesifikasi bonus yang dapat dilihat pada halaman berikutnya.

B01 - Cipher

Pada kenyataannya, password tidak pernah disimpan secara mentah. Password umumnya dan *best-practice*nya disimpan dalam keadaan *hashed* dan *salted*. Tetapi, saat anda mengkonfirmasi penggunaan *hash* dan *salt* pada sistem ini demi keamanan. Doni menolak karena ia tidak memerlukan sistem keamanan yang kuat dan ingin tidak terlalu rumit. Supaya tidak terlalu rumit ia meminta solusi keamanan dengan menggunakan *cipher*.

Algoritma cipher dibebaskan selama:

1. Kata semula tidak terlihat pada kata *ciphered*.
   Contoh yang tidak boleh:
   a. Menggunakan reversed, kata semula: password, kata *ciphered*: drowssap
   b. Menggunakan perantara di setiap huruf, kata semula: password, kata *ciphered*: p-a-s-s-w-o-r-d
   c. Dan sebagainya yang masih kelihatan.

   Contoh yang boleh (boleh pakai yang lain atau buat sendiri, ini hanya contoh)::
   Affine Cipher, definisinya untuk enkripsi `e(x) = (ax + b) mod m`
   Kunci dipilih: `a = 9 dan b = 3`
   Kata semula: password, Kata *ciphered*: idjjtzae *(Tidak terlihat kan?)*
2. Algoritma cipher memiliki key, jika key berbeda maka hasil enkripsi/dekripsi akan berbeda,
   Contoh memakai kasus Affine Cipher diatas,
   definisi untuk dekripsi $d(x) = a_{-1}(x - b) \bmod m$ dengan a inverse adalah *modular multiplicative inverse*
   - Kunci: `a = 7 dan b = 1`
     Kata *ciphered*: idjjtzae, Kata yang didapat dari dekripsi: beqqkwlt *(Beda karena key tidak tepat)*
   - Kunci: `a = 9 dan b = 3`
     Kata *ciphered*: idjjtzae, Kata yang didapat dari dekripsi: password
3. Dapat melakukan transformasi dua arah baik dari semula menjadi *ciphered* dan sebaliknya.

Untuk mengimplementasikan fitur ini, anda boleh (pilih salah satu opsi dibawah):

1. Berkreasi membuat desain dan algoritma sendiri.
2. Memakai algoritma yang sudah ada. Contoh: Affine Cipher, Keyed Caesar Cipher, dan lainnya.

selama tidak memakai *library python* atau eksternal lainnya.

Penyimpanan *key* untuk melakukan enkripsi dan dekripsi dapat di *hardcode* pada program. Akan tetapi, pastikan *key* bisa diubah untuk membuktikan nomor 2 di atas. Jangan lupa untuk menjelaskan di laporan bagaimana cara kerja cipher yang kalian buat atau gunakan.

B02 - Magic Conch Shell

"Binomo sebelum digitalisasi Magic Conch Shell"

Magic Conch Shell sebelum zaman digitalisasi masih berupa kerang dan sebuah tali yang dapat ditarik agar dapat berbicara. Namun, setelah zaman digitalisasi, Kerang ajaib ini diimplementasikan pada Binomo sehingga ia dapat menjawab pertanyaan secara ajaib. Indra sering sekali memakai fungsional ini untuk menentukan apakah ia akan melakukan *gacha* sekarang atau nanti.

Binomo akan meminta suatu input pertanyaan berupa string. Tidak perlu dilakukan validasi susunan bahasanya. Setelah itu, Binomo akan menjawab secara acak suatu jawaban, biasanya berupa "Ya", "Tidak", "Bisa Jadi", "Mungkin", "Tentunya", "Tidak mungkin", dan sebagainya, silakan kreasikan sendiri. Tidak perlu memastikan jawaban selalu konsisten (misal diberi pertanyaan **X** akan selalu menjawab **B**), jawab saja secara acak.

Untuk mengimplementasikan fitur ini tidak diperbolehkan untuk menggunakan `import random`. Implementasikan fitur ini dengan memanfaatkan `import time` dan *linear congruential generator (LCG)* untuk membangkitkan angka secara acak. Kata-kata respon yang mungkin dikeluarkan dibebaskan.

```
>>> kerangajaib
Apa pertanyaanmu? Apakah saya harus melakukan gacha sekarang?

Tidak.

# Kasus berulang menanyakan hal yang sama. Jawaban akan acak
>>> kerangajaib
Apa pertanyaanmu? Apakah saya harus melakukan gacha sekarang?

Mungkin.
```

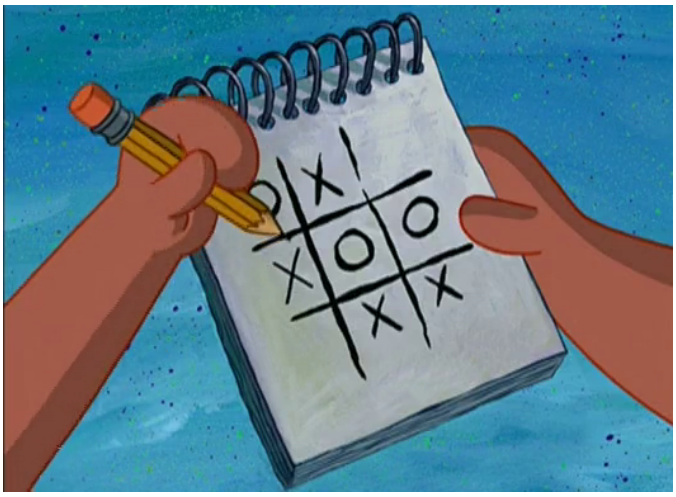
```
# Kasus diberi pertanyaan tidak masuk akal. Tidak perlu validasi.
>>> kerangajaib
Apa pertanyaanmu? Berapa 1+1, binomo?

Iya.

>>> kerangajaib
Apa pertanyaanmu? halo.

Bisa jadi.
```

B03 - Game Tic-Tac-Toe



"Nostalgianya Doni saat bermain tic-tac-toe pada kertas"

Dahulu, Doni dan Indra hidup dengan damai bermain *tic-tac-toe* pada kertas. Namun, semuanya berubah saat harga kertas naik. Akhirnya, mereka menabung dan membeli Binomo. Setelah sekian lama, mereka pun tidak bermain *tic-tac-toe* lagi karena lebih banyak permainan lain yang menarik di Binomo sampai Binomo rusak. Selama Binomo rusak, Doni nostalgia dengan *tic-tac-toe*. Dan ia pun memiliki sesuatu ide karena melihat anda yang hebat dalam memprogram. Idenya adalah suatu tantangan yang berkaitan dengan masa lalunya. Yap, Ia pun memutuskan untuk membuat permainan sederhana *tic-tac-toe*. Ia merasa nostalgia akan permainan itu dan sangat bersemangat untuk melihat anda mengimplementasikan permainan tersebut secara digital..

Permainan tic-tac-toe adalah permainan papan yang dimainkan oleh 2 pemain. Papan permainan ini berukuran 3 x 3. Pemain pertama akan menggunakan simbol “X” dan pemain kedua akan menggunakan simbol “O”. Pemain dapat menandai simbolnya pada kotak kosong secara bergantian. Pemenang adalah pemain pertama yang berhasil membuat simbolnya berurut 3 kali secara vertikal, horizontal, dan diagonal. Permainan dapat dihitung seri apabila kotak sudah penuh dan tidak ada yang berhasil menang. Lakukan validasi terhadap input yang dimasukkan. Apabila input tidak valid, tanyakan sampai input valid.

Cara bermain, pemain “X” dan pemain “O” secara bergiliran bertukar melakukan input yang valid (kotak ada dan tidak terisi oleh simbol apapun)..

Untuk Input/Output dibebaskan. Namun, jika memerlukan contoh, bisa mengakses ke tautan berikut, https://pastebin.com/raw/pg0DTHcB.

Contoh Kasus Akhir Game:

```
XO#
XO#
X##
X menang secara vertikal.
Kemenangan vertikal berlaku pada kolom lain.

XXX
XO#
OO#
X menang secara horizontal.
Kemenangan horizontal berlaku pada baris lain.

XXO
XO#
O##
O menang secara diagonal.
Kemenangan diagonal berlaku pada diagonal yang sebaliknya.

XOX
XOO
OXX
Seri. Tidak ada yang menang.
```

Batasan

Dalam tugas ini, ada beberapa batasan yang wajib dipenuhi oleh aplikasi yang kalian buat.

1. Menggunakan bahasa **Python 3.8+**
2. Hanya boleh import `os, sys, math, time, argparse, datetime` dan modul-modul serta fungsi yang telah dibuat sendiri.
   Tidak Boleh memakai pandas dan sejenisnya!!!
3. Tidak boleh menggunakan library eksternal dalam bentuk apapun
4. Tidak boleh menggunakan `.split(), .sort()`
   Silakan mengimplementasikan sendiri.
   Pengecualian untuk lower() dan upper().
5. Hanya boleh memakai fungsi-fungsi bawaan dari *python* yang diajarkan di kelas.
   Contoh yang tidak boleh: method list seperti append, insert, pop, dsb.
   Silakan mengimplementasikan sendiri.
6. **Tidak boleh membuat temporary CSV sebagai tempat penyimpanan sementara.** Gunakan List dan operasi list yang dibuat sendiri. **CSV hanya boleh digunakan sebagai penyimpanan ketika dipanggil load dan save.**
7. 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.

```
# Contoh program kasar yang mungkin bisa kalian tiru strukturnya
# Anggap semua fungsi yang dipanggil merupakan fungsi yang sudah dibuat sendiri pada modul lain
# PERHATIAN! INI HANYA CONTOH! BOLEH BEBAS NAMUN HANYA BOLEH ADA 1 MAIN PROGRAM
MUser = [] # Misalnya, Matriks User untuk menyimpan

load(“user.csv”, MUser) # membaca user.csv dan memasukan ke matriks user

while True:
   Perintah = input(“>>> “)
   If (Perintah == “help”):
       help()
   Else If (Perintah == “login”):
       login(MUser) # memanggil prosedur login, input username & password dilakukan di dalam prosedur login
   Else If ( . . . ): # perintah lain
       ( . . . )
   Else If (Perintah == “exit”):
       keluar_program()
   Else:
       perintah_tidak_valid()
```

Dalam membangun aplikasi yang aman, kita tidak boleh mempercayai input dari user. Pastikan anda melakukan **validasi input** sebelum memasukkan data ke database. Namun pada tugas kali ini ada beberapa input yang sudah dijamin valid sehingga tidak perlu melakukan validasi pada kolom tersebut. Pada tugas ini juga dijamin **tidak ada titik koma (;) pada seluruh input**..

Tentu saja, kalian dilarang bekerja sama antar kelompok.

Question and Answer

Link untuk tanya jawab pada tautan berikut.
Gunakan **NIM@std.stei.itb.ac.id** untuk mengakses tautan tersebut.

Kelompok

Daftar kelompok dapat diakses pada tautan berikut.

Deliverables

1. Source code 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. **Bersih**
    Hanya mengandung bagian-bagian yang diperlukan
    c. Well-Commented
2. Laporan Tugas Besar dibuat dengan template sebagai berikut:
    a. Halaman Cover, berisi minimum Kode dan Nama Mata Kuliah, Nama Tugas, Nomor Kelas, Nomor Kelompok, NIM dan Nama Anggota Kelompok, Nama Sekolah (Sekolah Teknik Elektro dan Informatika), Perguruan Tinggi (Institut Teknologi Bandung), dan Tahun (2022).
    b. Halaman pernyataan kelompok yang berisi seperti berikut, disertai nama dan NIM masing-masing anggota kelompok:

    > *"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 2020/2021."*

    c. Daftar Isi.
    d. Daftar Tabel.
    e. Daftar Gambar.
    f. Deskripsi persoalan: berisi penjelasan kembali (dengan "bahasa sendiri") tentang persoalan yang akan diselesaikan. Misal: "*Tugas besar ini meminta untuk membuatkan program mengupas kentang. Ada fungsional … untuk …"*
    g. Daftar pembagian kerja anggota kelompok, berdasarkan fitur yang dibuat, meliputi desain, implementasi, dan testing.
    Khusus untuk "F01 - Dekomposisi, Abstraksi dan Generalisasi Pola", harus dilakukan oleh <u>semua anggota</u> dan <u>tidak perlu</u> dimasukkan ke tabel berikut..

    Contoh:
    **) bisa menjadi procedure/fungsi/bagian dari program utama dan bisa lebih dari 1*
    ***) bisa dikerjakan oleh lebih dari 1 orang, 1 orang bisa mengerjakan desain, kode, dan test.*

Fitur	Implementasi *)	NIM Desainer **)	NIM Coder **)	NIM Tester **)
F02-Register	procedure tambah_user	16521600	16521601 16521603	16521602
F03-Login		16521601 16521603	16521603	16521602
F04-		16521602	16521601	16521600

			16521602	
...		...	...	...

h. Checklist hasil rancangan, implementasi dan testing setiap primitif.
   Contoh checklist:
   Untuk "F01 - Dekomposisi, Abstraksi dan Generalisasi Pola" <u>tidak perlu</u> dimasukkan ke tabel berikut.

Fitur	Desain	Implementasi	Testing
F02 - Register	V	V	V
F03 - Login	V	X	-
F04 - Menambah Game	-	-	-
...	...	...	...

Keterangan: V: sudah selesai dikerjakan, X: dikerjakan, tapi belum selesai, -: tidak dikerjakan sama sekali.

i. Desain command untuk setiap primitif (berisi: nama command, masukan, dan keluaran).
j. Desain kamus data (dalam notasi algoritmik).
k. Desain dekomposisi algoritmik dan fungsional program.
l. Spesifikasi untuk tiap modul/prosedur/fungsi yang dibuat (dalam notasi algoritmik).
m. Berikan screenshot hasil pengujian program berdasarkan fitur-fitur pada spesifikasi. Pada setiap fitur, screenshot minimal berisi: (1) data masukan, (2) data keluaran untuk input yang valid dan tidak valid jika terdapat validasi.
n. Lampiran: Hasil scan form asistensi.