Spesifikasi Tugas Besar
IF2110 Algoritma dan Struktur Data

Tim Asisten IF2110 Algoritma dan Struktur Data 2022/2023

Versi : **8**
Tgl. Revisi Terakhir : **31 Oktober 2022**

Deadline : **Minggu, 20 November 2022 12.00 WIB**

Revisi:

- 21 Oktober 2022: [Versi 2] Penambahan Batasan
- 22 Oktober 2022: [Versi 3] Penambahan informasi pada spesifikasi *inventory* dan penjelasan tentang root *tree recipe* yang tidak hanya satu.
- 25 Oktober 2022; [Versi 4] Penambahan Saran
- 27 Oktober 2022: [Versi 5] Update daftar isi & fix formatting
- 28 Oktober 2022; [Versi 6] Update mengenai validasi dan *error handling*
- 28 Oktober 2022: [Versi 7] Revisi contoh tampilan pengolahan makanan
- 31 Oktober 2022: [Versi 8] Tambahan informasi pada *command* undo/redo dan update penjelasan mengenai notifikasi, serta Auto-BNMO

o

Daftar Isi

Latar Belakang

"BNMO sedang memasak mengikuti program simulasi yang telah direkam Doni"

BNMO (dibaca: Binomo) adalah sebuah robot *game* milik Indra dan Doni. Akhir-akhir ini, Indra baru saja menjalin hubungan spesial dengan perempuan bernama Siska Kol. Dan dalam dekat waktu, Indra akan mengajak Siska Kol ke rumah untuk makan malam bersama Doni dan BNMO. Oleh karena itu, Indra meminta bantuan BNMO dan Doni untuk membantu mempersiapkan makan malam spesial tersebut. Saat itu juga, BNMO langsung tertarik untuk mengerjakan bagian masak karena ia sangat sering melihat video memasak di aplikasi toktok dan sangat terngiang-ngiang dengan *"mari kita cobaaa"*.

Namun, ada masalah. BNMO tidak tahu cara memasak dan Doni tidak bisa membantu persiapan karena ada hal lain. BNMO tidak bisa belajar dari video *youcub* karena BNMO adalah sebuah komputer sehingga hal yang paling mudah untuk dilakukan adalah membuatkan program simulasi untuk ditiru BNMO. Oleh karena itu, Doni meminta bantuan kalian untuk membuatkan program simulasi tersebut.

Spesifikasi Program

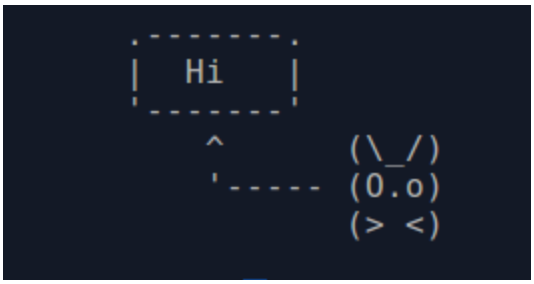
Buatlah sebuah program simulasi berbasis **CLI** (*command-line interface*). Program ini dibuat dalam bahasa C dengan menggunakan struktur data yang sudah kalian pelajari di mata kuliah ini. Kalian boleh menggunakan struktur data yang sudah kalian buat untuk praktikum pada tugas besar ini.

Kebutuhan Program

1. Inisiasi

a. Pada saat program pertama kali dijalankan, pengguna dapat menginisiasi aplikasi atau keluar dari aplikasi. Terdapat *splash screen* yang menandakan program berjalan dan muncul sebelum program meminta *command* pertama (START/EXIT) kepada pengguna.

Contoh *splash screen* dalam CLI:



Note:

- Boleh buat sendiri, atau cari ASCII Art yang tersedia di internet.
- Single line text juga diperbolehkan, seperti “Selamat datang di program simulasi masak!”
- Jika ASCII Art, ukuran dibebaskan.
- Bentuk juga dibebaskan sesuai kreativitas kelompok masing-masing.

b. *Command-command* pada fase ini adalah sebagai berikut:

i. **START**

Memulai aplikasi dan menginisiasi aplikasi dengan membaca *file* eksternal yang berisi konfigurasi aplikasi.

ii. **EXIT**

Keluar dari aplikasi.

c. Setelah *command* START dijalankan, aplikasi akan memiliki seluruh informasi yang dibutuhkan, seperti *layout* peta dan isinya, informasi makanan yang valid, dan pohon resep yang valid.

2. Simulator

a. Simulator merupakan representasi BNMO pada aplikasi.
b. Simulator memiliki beberapa informasi minimal yang harus disimpan, di antaranya:
    i. Nama pengguna simulator
    ii. Lokasi simulator di peta saat ini
    iii. *Inventory* makanan yang disimpan
c. Simulator dapat digerakkan pada peta sesuai yang akan dijelaskan pada bagian **Peta**.
d. Simulator dapat melakukan aksi-aksi tertentu sesuai tempat yang *adjacent* terhadapnya, aksi-aksi ini dijelaskan pada bagian **Pengolahan Makanan**.

3. Makanan

a. Makanan merupakan apapun yang bisa dimasak dan dimakan baik bahan mentah maupun matang.
b. Struktur data Makanan terdiri dari id makanan, nama makanan, waktu kedaluwarsa, lokasi aksi makanan, dan lama pengiriman makanan.
c. Waktu kedaluwarsa memanfaatkan struktur data Waktu/Time yang terdiri atas hari, jam dan menit.
d. Lokasi aksi makanan merupakan lokasi tempat eksekusi makanan seperti pemotongan, penggorengan, pembelian, dsb, dapat dilakukan. Contoh : Ayam Mentah memiliki lokasi aksi di tempat telepon setelah dilakukan pembelian (**BUY**) sedangkan Ayam Goreng memiliki lokasi aksi di tempat menggoreng setelah dilakukan aksi penggorengan (**FRY**).
e. Lama pengiriman makanan (delivery time) merupakan waktu yg dibutuhkan oleh simulator untuk mendapatkan makanan pada inventory setelah melakukan pembelian. Penjelasan tambahan dapat dilihat pada bagian **Pemesanan Bahan Makanan dan *Delivery*.**
f. Makanan yang valid didefinisikan terlebih dahulu pada file konfigurasi aplikasi.

4. Resep

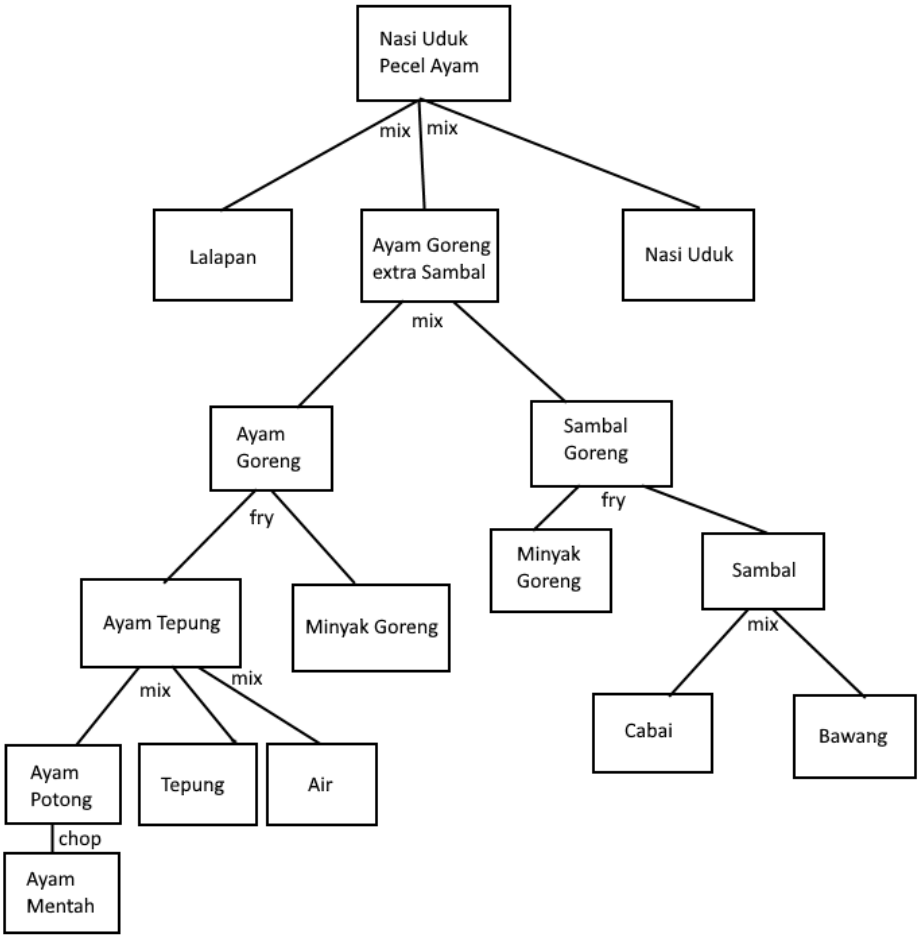
a. Resep merupakan kumpulan makanan yang direpresentasikan dengan *N-ary tree*.

b. Daftar resep yang valid didefinisikan pada konfigurasi *file*.

c. Setiap parent makanan dapat beralokasisal dari satu maupun lebih node makanan yang menghasilkan suatu makanan baru. Untuk detail setiap operasi yang dapat dilakukan untuk dapat menghasilkan makanan baru akan dijelaskan pada **Pengolahan Makanan**.

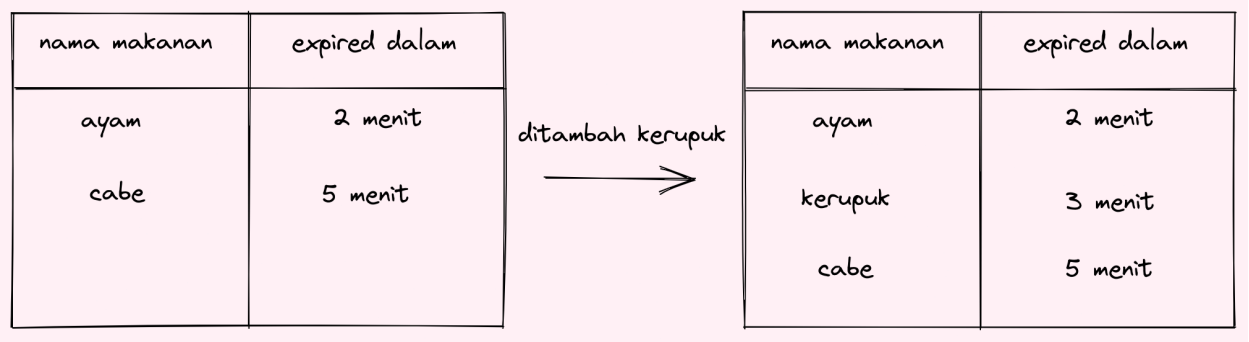
d. Root *Tree Recipe* tidak hanya satu. Silakan simpan di sebuah ADT array list statik. *Tree Recipe* tidak bisa diubah selama program berjalan. Hanya bisa diubah melalui file konfigurasi.

e. Contoh resep adalah sebagai berikut:



5. Inventory

a. Terdapat *inventory* yang dapat digunakan untuk mengecek makanan yang dipunyai.
b. Karena semua makanan memiliki atribut waktu kedaluwarsa, maka inventory yang dibuat menggunakan struktur data priority queue, semakin dekat suatu makanan dengan waktu kedaluwarsanya, maka semakin tinggi prioritasnya.
c. Jika suatu makanan sudah melewati batas expired, maka makanan akan langsung dikeluarkan otomatis dari inventorynya.
d. Inventory diimplementasikan dengan struktur data **priority queue dengan pendekatan list dinamik**. Prinsip priority queue diperlukan untuk proses penyimpanan makanan dengan *expiry* agar lebih mudah. Sedangkan pendekatan list dinamik diperlukan agar mempermudah proses *dequeue*.

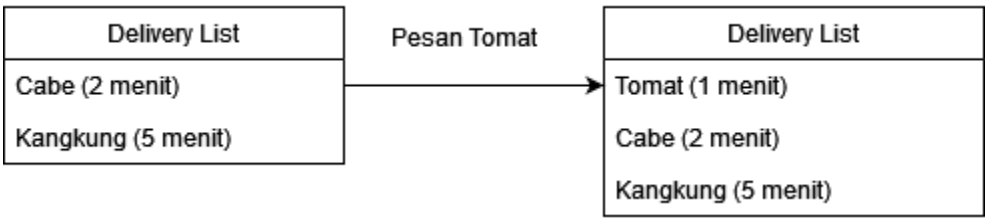


e. *Inventory* akan dianggap sebagai **Prio Queue** saat melakukan insert (dengan *enqueue*) dan *dequeue* (jika Head memiliki expired kurang dari sama dengan 0). Tetapi tidak menutup kemungkinan diperlakukan seperti array list untuk mencari di elemennya dan melakukan *remove* element dari queue (jangan lupa untuk dimajukan).

6. Pemesanan Bahan Makanan dan *Delivery*

a. Untuk melakukan pemesanan bahan makanan, gunakan *command* **BUY**.
b. Pemesanan hanya dapat dilakukan ketika simulator berada secara *adjacent* dengan “telepon” di peta.
c. Terdapat sebuah delivery time untuk pemesanan masing-masing bahan, pada saat delivery time sudah mencapai 0, bahan makanan yang dipesan akan masuk pada *inventory* pemain.

d. Simulator dapat melihat pemesanan makanan dengan *command* **DELIVERY**.
e. Jumlah bahan makanan yang dapat dibeli diasumsikan tidak terbatas.
f. Implementasi daftar pesanan menggunakan struktur data **priority queue.** Contoh sebagai berikut:



7. Peta

a. Peta bertujuan untuk menunjukkan navigasi BNMO dari satu tempat ke tempat lainnya untuk tujuan-tujuan tertentu.
b. Peta berukuran N x M dengan ukuran minimal 10 x 10 dengan titik (i, j) merupakan sebuah petak pada baris ke-i dan kolom ke-j.
c. Ukuran peta akan didefinisikan melalui file yang dibaca.
d. *Command* pada peta yang valid: **MOVE NORTH** / **MOVE EAST** / **MOVE WEST** / **MOVE SOUTH** yang masing-masing dapat menggerakkan BNMO sebanyak satu satuan sesuai dengan arah mata angin.
e. Simulator (S) tidak dapat menabrak atau melewati titik lokasi suatu tempat pada peta termasuk tempat telepon (T), tempat *mixing* (M), tempat menggoreng (F), tempat memotong (C), dan tempat merebus (B). Jika BNMO/Simulator (S) ingin memakai fitur dari suatu tempat, cukup cek adjacent dari lokasi BNMO/Simulator (S) pada saat itu.
f. Berikut adalah ilustrasi peta 10 x 10 yang ditampilkan kepada simulator

```
* * * * * * * * * * * *
* S                     *
*         T     X       *
*   M           X       *
*               X       *
*         X X X X       *
*   X                   *
*   X               C   *
*   X X X     F         *
*                       *
```

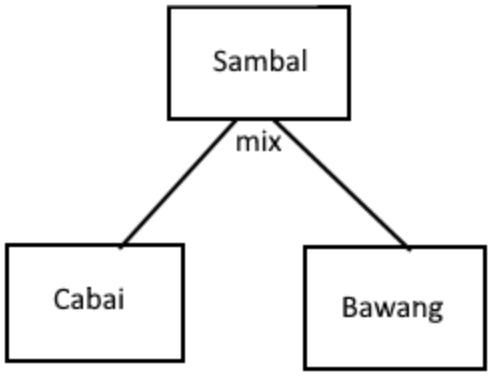
Peta diberi batasan berupa simbol * dan simulator tidak dapat melewati batasan tersebut. Karakter pada peta melambangkan lokasi-lokasi dari tempat pada peta tersebut. Berikut adalah legenda dari peta:

- Karakter 'S' adalah lokasi simulator atau BNMO saat ini berada.
- Karakter 'T' adalah lokasi telepon tempat BNMO dapat memesan bahan makanan.
- Karakter 'M' adalah lokasi tempat BNMO dapat melakukan *mixing* bahan makanan.
- Karakter 'C' adalah lokasi tempat BNMO dapat melakukan pemotongan bahan makanan.
- Karakter 'F' adalah lokasi tempat BNMO dapat menggoreng bahan makanan.
- Karakter 'B' adalah lokasi tempat BNMO dapat merebus bahan makanan.
- Karakter 'X' adalah tembok yang tidak bisa dilewati oleh BNMO.

8. Pengolahan Makanan

Terdapat beberapa operasi aksi yang dapat dilakukan terhadap makanan atau bahan makanan. Aksi-aksi tersebut adalah sebagai berikut:

a. **MIX**
   i. MIX merupakan *command* untuk mencampurkan dua atau lebih makanan menjadi satu makanan.
   ii. *Command* ini hanya dapat dilakukan di lokasi *mix* (M) pada peta.
   iii. Contoh:

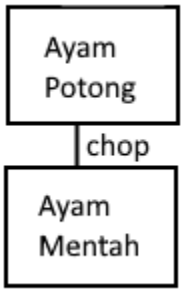


b. **CHOP**

i. CHOP merupakan *command* untuk memotong satu bahan makanan menjadi bahan yang sudah terpotong.

ii. *Command* ini hanya dapat dilakukan di lokasi *chop* (C) pada peta.

iii. Contoh:

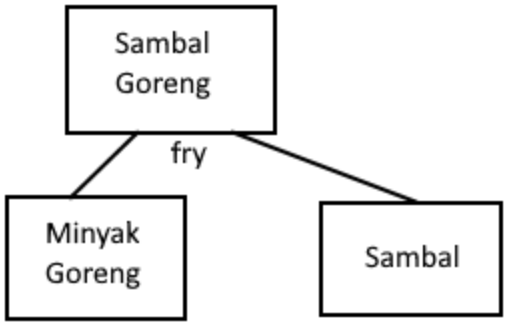


c. **FRY**

i. FRY merupakan *command* untuk menggoreng satu atau lebih bahan makanan. *Command* ini selalu memerlukan minyak goreng pada *inventory* (atau bahan lain tergantung dari *tree* resep yang didefinisikan).

ii. *Command* ini hanya dapat dilakukan di lokasi *fry* (F) pada peta.

iii. Contoh:

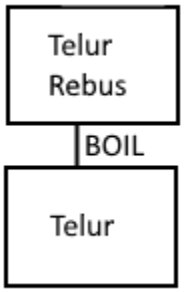


d. **BOIL**

i. BOIL merupakan *command* untuk merebus satu bahan makanan..

ii. *Command* ini hanya dapat dilakukan di lokasi *boil* (B) pada peta.

iii. Contoh:



9. Mekanisme Waktu

a. Pada saat aplikasi berjalan, setiap *command* perpindahan yang dilakukan oleh BNMO dan setiap *command* yang terkait pengolahan makanan (BUY, MIX, CHOP, FRY, BOIL), waktu pada aplikasi akan bertambah satu menit. Keberjalanan waktu ini kemudian dapat mengurangi waktu kedaluwarsa dari makanan yang disimpan. Selain itu, keberjalanan waktu ini juga mengurangi waktu *delivery* dari makanan yang dipesan.

b. Simulator dapat menunggu waktu tertentu dengan *command* **WAIT x y** dengan `x` adalah jam dan `y` adalah menit untuk diam tidak melakukan apa-apa dan hanya menambah waktu.

10. *Command-command* Lain

Di bawah ini adalah *command-command* lain yang tidak disebutkan di atas.

a. **UNDO / REDO**

Karena aplikasi ini merupakan simulasi, simulator dapat melakukan kesalahan. *Command* UNDO bertujuan untuk membatalkan *command* yang dilakukan oleh simulator dan mengembalikan *state* aplikasi ke sebelum *command* itu dieksekusi. Aplikasi juga dapat melakukan *command* REDO untuk membatalkan *command* UNDO.

Command undo/redo juga berlaku apabila makanan selesai diantar atau kadaluwarsa.

b. **CATALOG**

Command ini bertujuan untuk menampilkan bahan dan makanan yang tersedia pada aplikasi termasuk informasi detailnya.

c. **COOKBOOK**

Command ini bertujuan untuk menampilkan resep-resep yang tersedia pada sistem.

11. Validasi dan *Error Handling*

a. Untuk pembacaan file konfigurasi, isi file akan selalu dianggap valid dan tidak perlu ada validasi maupun *handling* apabila terjadi ketidakcocokan atau *error* lainnya selama isi file masih sesuai ketentuan dan memuat informasi minimal yang dapat digunakan aplikasi. Validasi dan *error handling* untuk pembacaan file bersifat opsional.

b. Selain pembacaan file, validasi dan *error handling* selama keberjalanan program **wajib** diimplementasi. Sebagai contoh: *command* yang tidak valid, *stack* undo kosong, lokasi pengolahan makanan yang tidak sesuai, dan lainnya.

12. Notifikasi

a. Notifikasi berguna untuk memberikan informasi atas suatu *event* kepada simulator agar simulator tahu apa yang sedang terjadi pada programnya.

b. Setiap *event* yang terjadi pada program **wajib** ditampilkan pada tampilan program untuk memberitahu pengguna bahwa *event* tersebut terjadi.

c. *Event-event* tersebut di antaranya:

   i. Makanan sudah masuk ke *inventory* dari *delivery*

   ii. Makanan pada *inventory* sudah kedaluwarsa

iii. *Command* undo dan redo dieksekusi

d. Contoh untuk *event* pada *command* undo:
    i. Simulator membatalkan eksekusi FRY dengan *command* undo, maka tampilkan notifikasi "Penggorengan makanan X dibatalkan".
    ii. Makanan selesai diantar / kedaluwarsa pada suatu waktu dan simulator mengeksekusi *command* undo, maka tampilkan notifikasi "Makanan X dihapus dari *inventory*" atau "Makanan X dikembalikan ke *inventory*".
    iii. Makanan sudah kedaluwarsa dan dihapus dari *inventory*, tampilkan notifikasi "Makanan X sudah kedaluwarsa".
e. Hal yang sama juga terjadi pada saat *command* redo dieksekusi.
f. Kalimat-kalimat notifikasi dibebaskan asal memberikan informasi yang jelas.

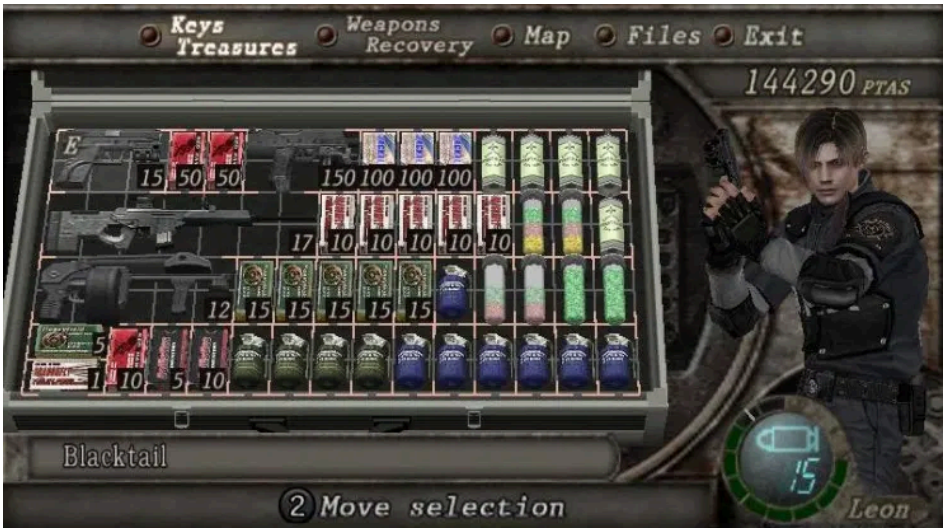
Bonus

Terdapat beberapa bonus implementasi pada aplikasi. Implementasi ini **tidak wajib dikerjakan**, tetapi *nice to have* dan dapat menambah penilaian aplikasi Anda. Silakan prioritaskan pengerjaan *requirements* wajib terlebih dahulu sebelum memutuskan untuk mengerjakan bonus.

1. Kulkas

Pengguna dapat meminta BNMO untuk menyimpan makanan di kulkas. Makanan yang ditaruh di kulkas akan dihilangkan dari PrioQueue/Inventory. Expired Time Makanan selama di Kulkas tidak berkurang. Peletakkan makanan pada kulkas harus diletakkan dalam sebuah matriks, lihat *gambar* referensi. Untuk ukuran setiap makanan didefinisikan pada ADT Makanan dengan representasi dua buah integer (NxM). Silakan mengubah file konfigurasi juga untuk perihal ukuran makanan. Makanan **tidak bisa** ditumpuk (*gambar referensi ditumpuk ada tulisan angka amount*, kalau kulkas BNMO tidak bisa). Makanan yang dimasukkan posisinya tidak boleh saling beririsan pada sel yang sama. Ukuran kulkas minimal 10x20 (10 vertikal, 20 horizontal). Untuk cara memasukkan dibebaskan kepada kelompok masing-masing, pastikan dijelaskan saat demo.

Referensi:



Sumber: Sistem Inventory Koper Resident Evil 4

2. Waktu Pengolahan Makanan

Secara *default*, pengolahan makanan dengan berbagai *command* yang sudah dijelaskan semuanya dijalankan dengan satu menit. Pada bonus ini, Anda diminta menambahkan waktu yang dibutuhkan untuk memproses makanan tersebut. Sebagai contoh, untuk membuat ayam goreng, diperlukan ayam dan minyak goreng yang digoreng selama 3 menit. Hanya mengubah penambahan waktu setiap kali melakukan aksi menggoreng, membeli, merebus, dan *mixing*.

3. Rekomendasi Makanan

Dari makanan yang disimpan BNMO, buatlah sebuah *command* yang dapat digunakan untuk menampilkan rekomendasi makanan yang dapat dibuat. Sebagai contoh, BNMO memiliki cabai x1, minyak goreng x2, dan ayam mentah x1, maka rekomendasi yang dapat diberikan adalah ayam potong, ayam goreng, sambal goreng, dan ayam geprek. Implementasi bonus ini wajib menggunakan struktur data Set. Hint: Gunakan Set dan prinsip Subset. Besar Set selalu linear dengan jumlah jenis makanan yang ada pada konfigurasi.

Contoh, lihat gambar berikut.

Inventory	
Minyak Goreng	1
Ayam Goreng	0
Ayam Potong	1
Ayam Tepung	1
Tepung	0
Cabai	0
Ayam Mentah	0
Beras	1

Resep Nasi	
Minyak Goreng	0
Ayam Goreng	0
Ayam Potong	0
Ayam Tepung	0
Tepung	0
Cabai	0
Ayam Mentah	0
Beras	1

Resep Ayam Tepung	
Minyak Goreng	0
Ayam Goreng	0
Ayam Potong	1
Ayam Tepung	0
Tepung	1
Cabai	0
Ayam Mentah	0
Beras	0

Resep Ayam Goreng	
Minyak Goreng	1
Ayam Goreng	0
Ayam Potong	0
Ayam Tepung	1
Tepung	0
Cabai	0
Ayam Mentah	0
Beras	0

Misalkan, di inventory terdapat 5 minyak goreng, 2 ayam potong, 1 ayam tepung, dan 1 beras. Maka representasi Setnya adalah seperti gambar di atas. Terdapat pula beberapa resep contoh yang dipakai sebagai contoh pada ilustrasi di atas.

- Rekomendasi yang akan muncul adalah: Nasi dan Ayam Goreng.
- Resep Nasi subset dari Inventory dan Resep Ayam Goreng subset dari Inventory.
- Ayam Tepung **tidak direkomendasikan** karena inventory tidak ada tepung sehingga Resep Ayam Tepung **bukan subset dari inventory.**
- Pada ilustrasi di atas, karena bahan yang didefinisikan hanya ada 8, maka panjang Set untuk inventory dan resep apapun selalu 8.
- Urutan set akan selalu sama. Kalau diambil contoh seperti ilustrasi seperti diatas, minyak goreng selalu indeks 0, dan seterusnya.
- Diperbolehkan menggunakan prinsip **union** untuk mengimplementasi rekomendasi bertingkat. Misalnya, di inventory hanya ada ayam mentah, tepung dan minyak goreng. Resep Ayam Goreng membutuhkan Ayam Tepung dan Minyak Goreng. Resep Ayam Goreng **di union** dengan Ayam Tepung. Ayam Tepung membutuhkan Ayam Potong dan Tepung. Resep yang tadi **di union** lagi dengan Ayam Tepung. Ayam Tepung membutuhkan Ayam Potong. Maka **di union** lagi dengan Ayam Potong. Karena hasil **union**nya merupakan **subset** dari inventory, maka rekomendasi bertingkat tersebut direkomendasikan. Hint: Bisa di iterasi dari pohon paling atas, jika tidak subset,

di union dengan barang yang tidak ada di inventory, dan seterusnya. (**Union-Find Algorithm**)

4. Auto-BNMO

Note: bonus nomor 4 memang *built different*. Prerequisite untuk mengerjakan spesifikasi bonus 4 ini mencakup beberapa mata kuliah lain. Jadikan sebagai tantangan untuk dicoba saja kalau ada waktu. Utamakan spesifikasi wajib.

Buatlah intelegensi buatan untuk BNMO merancang aksi-aksi yang harus dilakukan untuk memasak masakan yang diinginkan. Input berupa makanan yang ingin dimasak. Output merupakan langkah BNMO dalam mencapai state tersebut. Aturan makanan seperti *expired time* dan *delivery time* berjalan seperti biasa. Bisa menganggap kulkas tidak ada.

Aturan lain:

- Makanan yang diminta tidak boleh *expired* di tengah jalan. Apabila *expired*, harus dibuat lagi.

  Misal, diminta untuk membuat A dan B. Auto-BNMO membuat A lalu membuat B dan pada saat pembuatan B, A *expired*. Maka *goals* tidak tercapai. Auto-BNMO harus membuat A lagi.
- Jika permintaan tidak mungkin untuk dilakukan, Auto-BNMO dapat mengeluarkan: *hal tersebut tidak mungkin dilakukan*. Contoh permintaan yang tidak mungkin: waktu pembuatan A dan B sangat lama dan lebih kecil dari *expired time* keduanya. Sehingga, saat membuat A, B akan *expired* dan sebaliknya.
- Hint: Implementasikan rekomendasi bertingkat pada bonus 1, lalu buat *path finding*, dan *scheduling*. Paling sulit di bagian *scheduling*. Tidak bisa diselesaikan hanya dengan **if-else sederhana** atau **brute force** (mencoba semua kemungkinan).
- Auto-BNMO mengkalkulasi dari awal apakah sebuah task mungkin dijalankan. Contoh: Apabila membuat Ayam Goreng membutuhkan 1 jam (total jalan, masak, beli bahan, dsb.) dan expired timenya 30 menit. Dan Auto-BNMO disuruh membuat 2 Ayam goreng. Maka, langsung keluarkan pesan tidak mungkin untuk dilakukan.

- Apabila, ada beberapa cara untuk memasak, Auto-BNMO boleh memilih salah satu cara. Tetapi, tidak boleh memilih cara yang tidak mungkin dilakukan. Misalkan, Auto-BNMO harus memasak A, B dan C. Akan ada banyak urutan memasak tergantung dari jumlah node yang harus dilewati ketiga resep tersebut. Pilihlah salah satu cara yang mungkin untuk dilakukan. Kecuali semua kombinasi tidak mungkin sama sekali, maka refer ke pernyataan sebelum ini.
- Pilihlah algoritma *pathfinding*, *scheduling*, dan *constraint satisfaction problem* yang baik. Struktur Data yang sudah kalian pelajari dan Algoritma *searching* dan *sorting* bisa jadi bagian dari 3 permasalahan algoritma tersebut.
- Referensi (boleh pake yang lain, ini cuman referensi)
    a. Pathfinding: A Star, Djikstra
    b. Scheduling: Shortest Job Next, Priority Based Scheduling, Round Robin Scheduling, Multiple-Level Scheduling
    c. Constraint Satisfaction Problem: Backtracking
    d. Other: Breadth First Search
- **Note: Ingat ya, ini cuman tantangan, jangan dianggap serius sampe ngorbanin kesehatan kalian/tugas besar matkul lain. Kalau tidak sanggup, tidak apa-apa, emang bonus yang bagian ini berlebihan. Cuman bagus aja ketika kalian ada waktu luang lebih, bonus ini bisa jadi media kalian eksplor-eksplor biar semester depan ada gambaran di mata kuliah lain. Sekalian juga, biar kerasa kegunaan dari algoritma struktur data dan korelasinya dengan mata kuliah semester yang akan datang.**

Contoh Tampilan Aplikasi

Berikut adalah beberapa contoh referensi tampilan aplikasi. Anda dibebaskan untuk membuat tampilan aplikasi sesuai kreativitas masing-masing. Namun terdapat informasi-informasi yang wajib ada pada tampilan sesuai contoh berikut.

1. Tampilan *Idle*

```
BNMO di posisi: (0,0)
Waktu: 1.23
Notifikasi: -

* * * * * * * * * * * * *
* S                       *
*          T      X      *
*    M            X      *
*                 X      *
*          X X X X      *
*    X                  *
*    X                C  *
*    X X X      F        *
*                        *
*               B        *
* * * * * * * * * * * * *

Enter Command: _
```

2. Tampilan Pemesanan Makanan

```
BNMO di posisi: (1,2)
Waktu: 1.23
Notifikasi: -

* * * * * * * * * * * * *
*                       *
*      S    T      X      *
*    M            X      *
*                 X      *
*          X X X X      *
*    X                  *
*    X                C  *
*    X X X      F        *
*                        *
*               B        *
* * * * * * * * * * * * *
```

```
Enter Command: BUY

BNMO tidak berada di area telepon!
```

```
BNMO di posisi: (1,3)
Waktu: 1.23
Notifikasi: -

* * * * * * * * * * * *
*                       *
*        S T      X     *
*   M             X     *
*                 X     *
*           X X X X     *
*   X                   *
*   X               C   *
*   X X X      F        *
*                       *
*              B        *
* * * * * * * * * * * * *

Enter Command: BUY
```

```
======================
=         BUY        =
======================
List Bahan Makanan:
   1. Cabai (2 menit)
   2. Bawang (1 menit)
   3. Ayam Mentah (5 jam)

Kirim 0 untuk exit.

Enter Command: 1

Berhasil memesan Cabai. Cabai akan diantar dalam 2 menit.
```

```
======================
=         BUY        =
======================
List Bahan Makanan:
   1. Cabai (2 menit)
   2. Bawang (1 menit)
   3. Ayam Mentah (5 jam)

Kirim 0 untuk exit.

Enter Command: 0
```

```
BNMO di posisi: (1,3)
Waktu: 1.24
```

```
Notifikasi: -

* * * * * * * * * * * *
*                       *
*        S T      X     *
*   M             X     *
*                 X     *
*           X X X X     *
*   X                   *
*   X                C  *
*   X X X      F        *
*                       *
*               B       *
* * * * * * * * * * * * *

Enter Command: _
```

3. Tampilan Pengolahan Makanan

Pada contoh ini akan ditampilkan pengolahan makanan untuk *command* FRY.

```
BNMO di posisi: (7,5)
Waktu: 1.23
Notifikasi: -

* * * * * * * * * * * *
*                       *
*          T      X     *
*   M             X     *
*                 X     *
*          X X X X      *
*   X                   *
*   X                C  *
*   X X X    S F        *
*                       *
*              B        *
* * * * * * * * * * * * *

Enter Command: FRY
```

```
======================
=         FRY        =
======================
List Bahan Makanan yang Bisa Dibuat:
   1. ~~Ayam Potong~~ Ayam Goreng
   2. ~~Sambal~~ Sambal Goreng
```

```
Kirim 0 untuk exit.

Enter Command: 1

~~Gagal menggoreng ayam potong karena kamu tidak memiliki bahan berikut:~~
Gagal membuat ayam goreng karena kamu tidak memiliki bahan berikut:
   1. Ayam Potong

Enter Command: 2

~~Sambal selesai digoreng dan sudah masuk ke inventory!~~
Sambal goreng selesai dibuat dan sudah masuk ke inventory!

Enter Command: 0
```

```
BNMO di posisi: (7,5)
Waktu: 1.24
Notifikasi: -

* * * * * * * * * * * *
*                     *
*          T     X    *
*   M            X    *
*                X    *
*          X X X X    *
*   X                 *
*   X               C *
*   X X X   S F       *
*                     *
*             B       *
* * * * * * * * * * * *

Enter Command: _
```

4. Tampilan Melihat Makanan yang Ada

```
BNMO di posisi: (0,0)
Waktu: 1.23
Notifikasi: -

* * * * * * * * * * * *
* S                   *
*          T     X    *
*   M            X    *
*                X    *
*          X X X X    *
*   X                 *
```

```
*   X               C  *
*   X X X     F        *
*                      *
*             B        *
* * * * * * * * * * * * *

Enter Command: CATALOG

List Makanan
(nama - durasi kedaluwarsa - aksi yang diperlukan -
delivery time)
   1. Cabai - 3 jam - BUY - 2 jam
   2. Ayam Mentah - 5 menit - BUY - 4 jam
   3. Ayam Potong - 5 menit - CHOP - 0
   4. Ayam Goreng - 8 jam - FRY - 0
```

5. Tampilan Melihat Resep yang Ada

```
BNMO di posisi: (0,0)
Waktu: 1.23
Notifikasi: -

* * * * * * * * * * * * *
* S                      *
*          T     X      *
*   M            X      *
*                X      *
*          X X X X      *
*   X                    *
*   X               C  *
*   X X X     F        *
*                      *
*             B        *
* * * * * * * * * * * * *

Enter Command: COOKBOOK

List Resep
(aksi yang diperlukan - bahan...)
   1. Ayam Goreng
      FRY - Ayam Potong - Minyak Goreng
   2. Telur Rebus
      BOIL - Telur
   3. Sambal Goreng
      FRY - Sambal - Minyak Goreng
   4. Ayam Sambal
      MIX - Ayam Goreng - Sambal Goreng
```

6. Tampilan Melihat Makanan yang Ada di *Inventory*

```
BNMO di posisi: (0,0)
Waktu: 1.23
Notifikasi: -

* * * * * * * * * * * * *
* S                       *
*          T     X       *
*   M            X       *
*                X       *
*          X X X X       *
*   X                     *
*   X                C    *
*   X X X      F          *
*                         *
*              B          *
* * * * * * * * * * * * * *

Enter Command: INVENTORY

List Makanan di Inventory
(nama - waktu sisa kedaluwarsa)
   1. Ayam Goreng - 2 jam
   2. Ayam Goreng - 5 jam
   3. Cabai - 3 jam
   4. Bawang - 1 jam
```

7. Tampilan Melihat Makanan yang Sedang Diantar

```
BNMO di posisi: (0,0)
Waktu: 1.23
Notifikasi: -

* * * * * * * * * * * * *
* S                       *
*          T     X       *
*   M            X       *
*                X       *
*          X X X X       *
*   X                     *
*   X                C    *
*   X X X      F          *
*                         *
*              B          *
* * * * * * * * * * * * * *
```

```
Enter Command: DELIVERY

List Makanan di Perjalanan
(nama - waktu sisa delivery)
   1. Ayam Goreng - 2 jam 5 menit
   2. Ayam Goreng - 5 jam
   3. Cabai - 3 menit
   4. Bawang - 1 menit
```

8. Tampilan Ketika Bahan Makanan yang Dipesan Sudah Sampai

```
BNMO di posisi: (0,0)
Waktu: 1.23
Notifikasi: -

* * * * * * * * * * * *
* S                     *
*          T     X     *
*   M            X     *
*                X     *
*          X X X X     *
*   X                  *
*   X               C  *
*   X X X     F        *
*                      *
*              B       *
* * * * * * * * * * * *

Enter Command: DELIVERY

List Makanan di Perjalanan
(nama - waktu sisa delivery)
   1. Ayam Goreng - 2 menit
   2. Ayam Goreng - 5 menit
   3. Cabai - 3 menit
   4. Bawang - 1 menit
```

```
BNMO di posisi: (0,0)
Waktu: 1.23
Notifikasi: -

* * * * * * * * * * * *
* S                     *
*          T     X     *
*   M            X     *
*                X     *
*          X X X X     *
*   X                  *
*   X               C  *
```

```
*   X X X     F       *
*                     *
*             B       *
* * * * * * * * * * * *

Enter Command: WAIT 0 1
```

```
BNMO di posisi: (0,0)
Waktu: 1.24
Notifikasi:
   1. Bawang sudah diterima oleh BNMO!

* * * * * * * * * * * *
* S                     *
*         T     X       *
*   M           X       *
*               X       *
*         X X X X       *
*   X                   *
*   X               C   *
*   X X X     F         *
*                       *
*             B         *
* * * * * * * * * * * * *

Enter Command: DELIVERY

List Makanan di Perjalanan
(nama - waktu sisa delivery)
   1. Ayam Goreng - 1 menit
   2. Ayam Goreng - 4 menit
   3. Cabai - 2 menit
```

9. Tampilan Ketika Bahan Makanan Mencapai Waktu Kedaluwarsa

```
BNMO di posisi: (0,0)
Waktu: 1.23
Notifikasi: -

* * * * * * * * * * * *
* S                     *
*         T     X       *
*   M           X       *
*               X       *
*         X X X X       *
*   X                   *
*   X               C   *
*   X X X     F         *
```

```
*                     *
*             B       *
* * * * * * * * * * * *

Enter Command: INVENTORY

List Makanan di Inventory
(nama - waktu sisa kedaluwarsa)
   1. Ayam Goreng - 2 menit
   2. Ayam Goreng - 5 menit
   3. Cabai - 3 menit
   4. Bawang - 1 menit
```

```
BNMO di posisi: (0,0)
Waktu: 1.23
Notifikasi: -

* * * * * * * * * * * *
* S                     *
*         T     X       *
*   M           X       *
*               X       *
*         X X X X       *
*   X                   *
*   X               C   *
*   X X X     F         *
*                       *
*             B         *
* * * * * * * * * * * * *

Enter Command: MOVE EAST
```

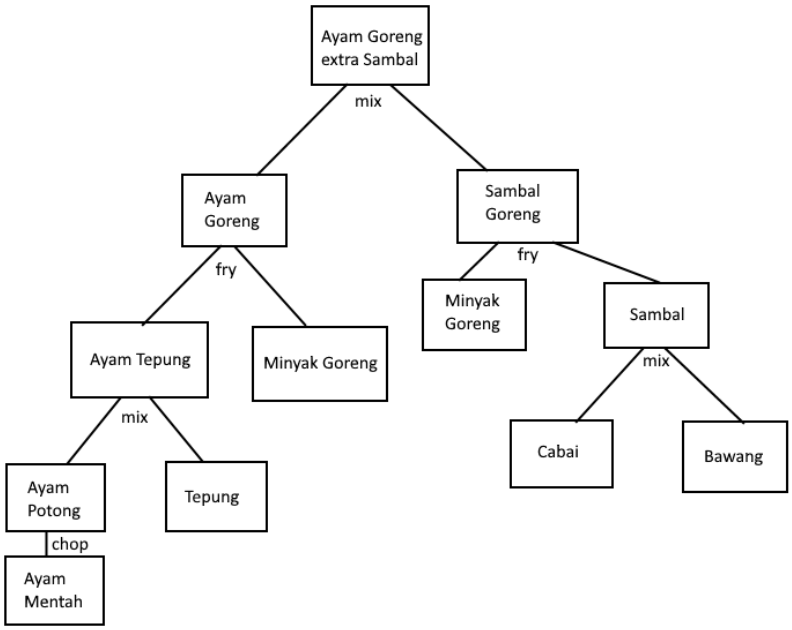
```
BNMO di posisi: (0,1)
Waktu: 1.24
Notifikasi:
   1. Bawang kedaluwarsa.. :(

* * * * * * * * * * * *
*   S                   *
*         T     X       *
*   M           X       *
*               X       *
*         X X X X       *
*   X                   *
*   X               C   *
*   X X X     F         *
*                       *
*             B         *
* * * * * * * * * * * * *

Enter Command: INVENTORY
```

```
List Makanan di Inventory
(nama - waktu sisa kedaluwarsa)
   1. Ayam Goreng - 1 menit
   2. Ayam Goreng - 4 menit
   3. Cabai - 2 menit
```

Contoh Konfigurasi Aplikasi



1. Konfigurasi Makanan

Konfigurasi disusun atas:

- Baris pertama, N makanan
- Baris selanjutnya dibaca, ID Makanan, Judul, Lama *Expired*, Waktu Pengiriman, Aksi yang harus dilakukan untuk mendapat makanan ini.
- ID bersifat *unique* tetapi tidak urut, tidak perlu validasi. ID ganda akan menimpa makanan yang sudah didefinisikan sebelumnya.
- Kalau bahan makanan tersebut didapat bukan dari **BUY**, waktu pengiriman bisa dibuat 0 0 0.

- Warna abu-abu adalah komentar, tidak usah di *parsing* atau hilangkan saat *copy* ke file.

```
6            # ini adalah N
11           # ID Ayam Mentah: 11
Ayam Mentah  # Judul makanan 1
1 0 0        # Akan expired dalam 1 hari
0 0 15       # Lama pengiriman 15 menit
Buy          # Aksi: harus dibeli
21           # ID Ayam Potong: 21
Ayam Potong  # Judul makanan 2
0 2 0        # Akan expired dalam 2 jam
0 0 0        # Bukan barang yang dibeli, tidak ada waktu kirim
Chop         # Aksi: harus dibuat dengan cara dipotong
10           # ID Tepung: 10
Tepung
0 1 0
0 0 30
Buy
15           # ID Minyak Goreng: 15
Minyak Goreng
0 1 30
0 1 0
Buy
31           # ID Ayam Tepung: 31
Ayam Tepung
0 0 30
0 0 0
Mix
37           # ID Ayam Goreng: 37
Ayam Goreng
0 0 15
0 0 0
Fry
```

2. Konfigurasi Resep

Konfigurasi disusun atas:

- Baris pertama, N resep
- Baris selanjutnya dibaca ID parent, M child, dan ID child sebanyak M.
- Warna abu-abu adalah komentar, tidak usah di *parsing* atau hilangkan saat *copy* ke file.
- Perlu di ingat, child bisa berjumlah berapapun. Implementasikan Nary Tree.

```
3  # Ada 3 resep
21 1 11
```

```
31 2 21 10
37 2 31 15
# Baris 2 (21 1 11) memiliki arti:
# Ayam Potong (ID: 21) memiliki child sebanyak 1
# dan childnya adalah Ayam Mentah (ID: 11)

# Baris 3 (31 2 21 10) memiliki arti:
# Ayam Tepung (ID: 31) memiliki child sebanyak 2
# dan childnya adalah Ayam Potong (ID: 21) dan Tepung (ID:
10)

# Baris 4 (37 2 31 15) memiliki arti:
# Ayam Goreng (ID: 37) memiliki child sebanyak 2
# dan childnya adalah Ayam Tepung(ID: 31) dan Minyak (ID: 15)
```

3. Konfigurasi Peta

Konfigurasi disusun atas:

- Baris pertama, ukuran peta, N x M
- Baris selanjutnya adalah representasi peta. # untuk petak kosong, sisanya sama seperti simbol peta yang sudah disebutkan sebelumnya.

```
10 10
S#########
####T##X##
#M######X##
#######X##
####XXXX##
#X########
#X######C#
#XXX##F###
##########
######B###

# Akan Menghasilkan Peta:
# * * * * * * * * * * *
# * S                 *
# *       T     X     *
# *   M         X     *
# *               X   *
# *         X X X X   *
# *   X               *
# *   X           C   *
# *   X X X     F     *
# *                   *
# *           B       *
# * * * * * * * * * * *
```

Daftar ADT yang Digunakan

Anda diwajibkan menggunakan ADT di bawah ini. Selain itu, Anda dapat pula menggunakan/membuat ADT lain. Namun, cantumkan analisis serta alasan kenapa menggunakan ADT tersebut pada laporan. Semua yang dicantumkan di bawah ini adalah wajib, tidak boleh disubtitusi dengan ADT yang lain. ADT lainnya hanya sebagai tambahan yang berfungsi untuk memperlengkap, menambah fitur, atau menyelesaikan bonus.

Note: Seluruh ADT yang dibuat wajib dibuatkan *driver*-nya.

1. ADT Sederhana

ADT ini digunakan untuk membentuk struktur data Point, Waktu, Makanan, dan Simulator.

a. ADT Point wajib memiliki koordinat X dan Y. ADT ini digunakan untuk merepresentasikan lokasi.
b. ADT Waktu wajib memiliki data Hari, Jam, dan Menit.
c. ADT Makanan wajib memiliki data ID makanan, Nama makanan, Waktu kedaluwarsa, Lokasi aksi, dan Lama pengiriman.
d. ADT Simulator wajib memiliki data Nama pengguna, Lokasi saat ini, dan Inventory.

2. ADT List Statik

ADT ini digunakan untuk menyimpan list makanan dan list resep yang ada pada aplikasi. List ini immutable (tidak dapat diubah) dan elemen pada list ini hanya boleh di-*copy*. List ini berfungsi untuk dijadikan acuan ketika pemesanan bahan makanan dan berisi data dari konfigurasi bukan data dari *state* aplikasi.

3. ADT Matriks

ADT ini digunakan untuk representasi peta dari program. Peta yang dibuat dalam bentuk karakter-karakter tertentu seperti contoh tampilan disimpan ke dalam sebuah ADT matriks.

4. ADT Mesin Karakter dan Mesin Kata

ADT ini digunakan untuk melakukan parsing *command* program dan pembacaan *file* konfigurasi ke dalam aplikasi.

5. ADT Queue dengan Pendekatan Array List Dinamik

ADT ini digunakan untuk menyimpan makanan yang dimiliki oleh simulator alias *inventory*. ADT Queue wajib dibuat dalam bentuk PrioQueue dengan pendekatan Array List Dinamik untuk mendukung penyimpanan makanan dengan *expiry* agar lebih mudah untuk diproses. Pendekatan List diperlukan untuk mempermudah proses *dequeue*.

6. ADT Stack

ADT ini digunakan untuk meng-*undo* atau meng-*redo command* yang dilakukan oleh simulator pada aplikasi.

7. ADT Tree

ADT ini digunakan untuk menyimpan resep. Jangan lupa untuk pakai N-ary Tree, karena jumlah childnya bisa tidak ada sampai berapapun. Gunakan pointer of pointer of Node pada struct Node agar bisa membuat *array of node pointer*. Diperbolehkan juga memakai Array List Dinamis dengan elemen pointer of Node pada Node Tree.

8. ADT Lain

Anda diperbolehkan menambahkan dan menggunakan ADT lainnya untuk mendukung program. Silakan dibuat dengan kreativitas masing-masing bila diperlukan atau jika ingin mengerjakan fitur ekstra dari kelompok atau untuk mengerjakan bonus. ADT Lain ini sifatnya opsional dan tidak boleh menjadi subtitusi ADT Wajib. ADT yang diwajibkan di atas tidak boleh disubtitusi dengan ADT buatan sendiri.

Batasan

Hanya diperbolehkan menggunakan library sebagai berikut.

- Math (untuk operasi matematika seperti pangkat, akar, dsb. jika diperlukan)
- Stdio (untuk melakukan print, open)
- Stdlib (untuk malloc dan free)

Gunakan **mesin karakter** dan **mesin kata** untuk melakukan manipulasi/operasi string. Jika tidak ada, implementasikan sendiri.

Saran

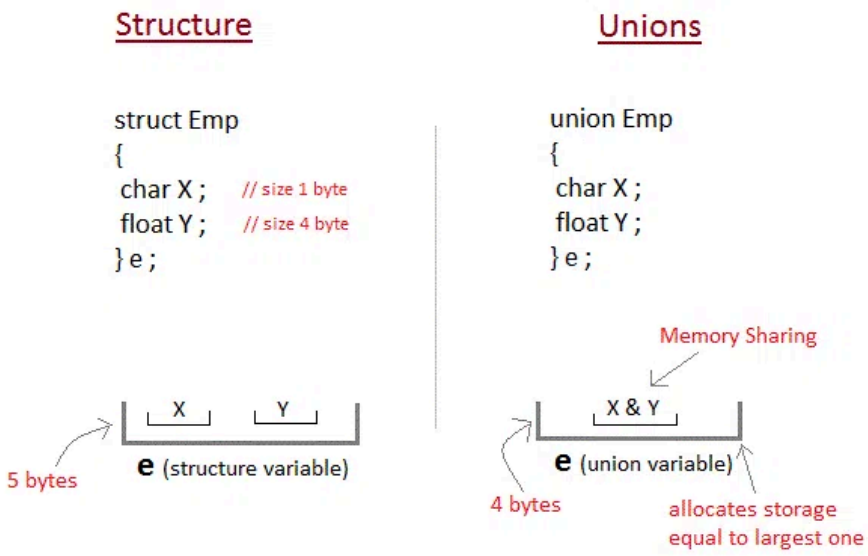
Pada bagian ini, hanya dituliskan **saran-saran** yang sekiranya bisa membantu mempermudah pengerjaan. **Sifat dari bagian ini adalah TIDAK WAJIB.** Silakan diikuti atas dasar kemauan kalian sendiri.

- Menggunakan **union**

  Union adalah tipe data yang dapat berisi sekumpulan tipe data lain yang dibungkus menjadi satu dan memiliki alamat memori yang sama. Hal ini bisa dimanfaatkan untuk menjadi tipe data ADT sehingga sebuah ADT dapat dipakai untuk beberapa tipe data tanpa pendefinisian ulang.

  Kenapa mempermudah? Kalian hanya perlu mengimplementasikan 1 ADT untuk semua tipe. Sebagai contoh, jika kalian mau sebuah ListDinamis bisa untuk integer dan float, normalnya diperlukan dua kali implementasi yaitu ListDinamisInteger dan ListDinamisFloat. Dengan union, kalian bisa membuat sebuah Union (sebut saja mawar) dan kalian hanya perlu implementasi ListDinamisMawar. Dan list dinamis tersebut dapat dipakai untuk integer dan float.

Secara teknis memori, dapat digambarkan sebagai berikut.



Sharing memory tersebut mengakibatkan union hanya bisa diisi satu field saja. Jika kalian assign nilai ke X pada union tersebut, nilai Y akan hilang, begitu juga sebaliknya. **Berbeda dengan struct**, jika assign X, nilai Y tetap ada.

Silakan gunakan Union untuk mendefinisikan Element Type saja, bukan ADTnya.

Contoh:

```
union Mawar {
   float X;
   int Y;
   char Z;
};

struct ListDinamisMawar {
   int N;
   Mawar* Array;
};
```

List Dinamis di atas dapat dipakai untuk float/integer/char. Isi dari union juga bisa berupa struct lain (misal Resep/Simulator/Game), tinggal definisikan saja seperti kalian mendefinisikan struct dalam struct. Jika memakai union, silakan diasumsikan, semua element union nya sama dalam sebuah ADT.