

LAPORAN TUGAS BESAR

IF2110/Algoritma dan Struktur Data

SIMULATOR MASAK


BNMO

Dipersiapkan oleh Kelompok G:

Yanuar Sano Nur Rasyid	13521110
William Nixon	13521123
Nicholas Liem	13521135
Akhmad Setiawan	13521164
Reza Pahlevi Ubaidillah	13521165

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		<i>IF2110-TB-G-2</i>		<i>34</i>
		<i>Revisi</i>	<i>1</i>	<i>19-11-2022</i>

Daftar Isi

1 Ringkasan.....	4
2 Penjelasan Tambahan Spesifikasi Tugas	4
2.1 Spesifikasi Kulkas	4
2.2 Spesifikasi Waktu Pengolahan Makanan	5
2.3 Spesifikasi Rekomendasi Makanan.....	5
3 Struktur Data (ADT)	6
3.1 ADT Point	6
3.2 ADT Time	6
3.3 ADT Makanan.....	6
3.4 ADT Peta.....	7
3.5 ADT List Statik	7
3.6 ADT Matriks	7
3.7 ADT Mesin Karakter.....	7
3.8 ADT Mesin Kata	7
3.9 ADT Word.....	8
3.10 ADT Priority Queue Dinamis	8
3.11 ADT Stack	8
3.12 ADT List Linier	9
3.13 ADT Set.....	9
3.14 ADT Tree.....	9
3.15 ADT Kulkas.....	9
4 Program Utama	10
5 Algoritma-Algoritma Menarik.....	11
5.1 Algoritma Undo-Redo.....	11
5.2 Algoritma Set-Tree Rekomendasi Makanan	11
6 Data Test	12
6.1 Compile Program	12
6.2 Start dan Exit Program	12
6.3 Help	13
6.4 Catalog	14
6.5 Cookbook	14
6.6 Move.....	15
6.7 Wait	17
6.8 Buy, Delivery, Boil, Fry, Chop, dan Mix.....	18
6.9 Inventory	22
6.10 Kulkas	23
6.11 Rekomendasi Makanan.....	26
6.12 Makanan Kedaluwarsa.....	27
6.13 Undo	27
6.14 Redo.....	28
7 Test Script	28
8 Pembagian Kerja dalam Kelompok	31
9 Lampiran	31
9.1 Deskripsi Tugas Besar 2.....	31
9.2 Notulen Rapat.....	31

9.3	Log Activity Anggota Kelompok.....	36
-----	------------------------------------	----

1 Ringkasan

Simulasi Masak BNMO merupakan program berbasis CLI (*Command-line Interface*) yang mensimulasikan bagaimana sebuah resep makanan dapat dibuat, dengan menjalankan *command* yang tersedia. Pada mulanya, program simulator memiliki beberapa bahan makanan sesuai konfigurasi yang ada, untuk kemudian dapat diolah sesuai dengan resep yang bersesuaian. Simulator dapat bergerak sepanjang peta untuk melakukan berbagai hal, mulai dari membeli tambahan bahan makanan, mengolah makanan seperti memotong, menggoreng, merebus, dan mencampurkan bahan makanan di tempat tertentu pada peta. Simulator harus berada di dekat tempat di mana makanan dapat diolah. Program dibuat menggunakan bahasa C dengan mengimplementasikan ADT yang sudah dipelajari pada mata kuliah IF2110 - Algoritma dan Struktur Data.

Berdasarkan deskripsi tersebut, program simulasi masak ini dibuat dengan memadukan berbagai komponen program seperti yang tercantum dalam laporan ini. Laporan ini berisi berbagai penjelasan mengenai spesifikasi fitur program simulator, penjelasan mengenai ADT yang digunakan, algoritma dari program utama, data-data dan *script* yang digunakan untuk menguji jalannya program, serta lampiran pembagian tugas, notulensi rapat, dan *log activity* anggota kelompok.

Kesimpulannya, program Simulasi Masak BNMO ini mulanya memiliki beberapa konfigurasi program, yang pertama terdapat konfigurasi makanan yang berisikan jumlah makanan yang tersedia, ID makanan, nama makanan, waktu kedaluwarsa, aksi yang harus dilakukan untuk mendapatkan makanan tersebut, serta berapa lama waktu pengiriman ketika makanan tersebut dibeli. Selain itu, terdapat pula konfigurasi resep yang membutuhkan implementasi *N-ary tree*, serta konfigurasi peta di mana nantinya program simulator dapat bergerak. Program juga memiliki beberapa fitur pendukung lainnya seperti *inventory* untuk menyimpan makanan, serta notifikasi untuk membantu melihat beberapa data seperti ketika suatu makanan telah kedaluwarsa dan ketika bahan makanan telah sampai dalam proses pengiriman. Program ini juga memiliki beberapa bonus fitur tambahan seperti kulkas yang dapat menyimpan makanan yang ada di *inventory*, waktu pengolahan makanan, serta rekomendasi makanan yang tersedia.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 Spesifikasi Kulkas

Kulkas yang diimplementasikan berupa matriks berukuran 10x20. Bahan makanan dapat disimpan dari *inventory* ke dalam kulkas untuk mempertahankan waktu kedaluwarsanya. Proses penyimpanan makanan tidak dapat ditumpuk dan orientasinya dipertahankan dari awal. Saat menyimpan bahan makanan, pengguna diminta untuk menginput lebar dan panjang dari bahan makanan yang ingin disimpan. Jika masih memungkinkan ruang untuk menyimpan bahan makanan, maka makanan akan disimpan dan diberi ID khusus. Jika tidak ada ruang, muncul pemberitahuan bahwa tidak dapat menyimpan tambahan makanan lagi. Ketika ingin mengambil makanan dari kulkas, pengguna diminta untuk menginput ID makanan yang ada pada kulkas untuk kemudian disimpan ke *inventory*.

2.2 Spesifikasi Waktu Pengolahan Makanan

Waktu pengolahan makanan diset berbeda-beda sesuai jenis makanannya, tidak disamaratakan. Jika makanan telah selesai diolah, akan muncul pemberitahuan bahwa makanan sudah tersedia dan telah masuk *inventory*.

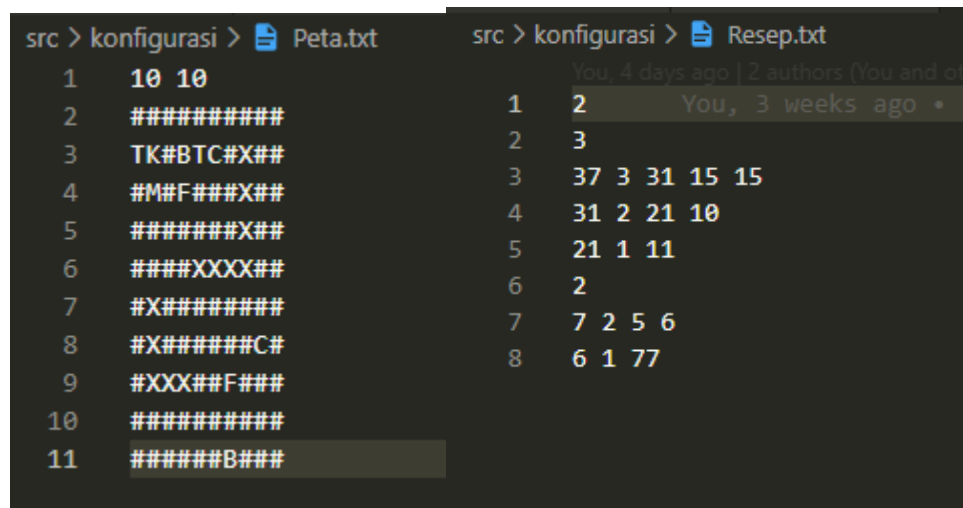
2.3 Spesifikasi Rekomendasi Makanan

Pengguna yang memiliki bahan makanan tertentu dapat mengetahui resep makanan yang direkomendasikan dan dapat diolah sesuai dengan bahan-bahannya. Fitur ini menggunakan struktur data *set* dan *tree*. Resep disimpan dalam bentuk *tree* dan kemudian menggunakan konsep subset untuk menentukan makanan yang dapat diolah. Besar set yang digunakan selalu linier dengan jumlah jenis makanan pada konfigurasi. Jika *inventory* kosong, maka tidak ada yang direkomendasikan.

2.4 Konfigurasi File

Terdapat 3 file konfigurasi yang harus diubah sebelum program dijalankan. Yaitu file konfigurasi makanan, resep, dan peta. Format konfigurasi resep dan peta dilakukan sesuai dengan spesifikasi tugas besar, sedangkan pada resep dilakukan sedikit modifikasi. Pada peta dibutuhkan dimensi dari peta tersebut diikuti dengan layout peta beserta stationnya. Pada makanan dibutuhkan banyak makanan, disusul dengan ID, nama, durasi pengiriman, durasi expired, dan command untuk mendapatkannya.

Pada resep, dibutuhkan banyak root node, disusul dengan konfigurasi dari masing-masing root node, yaitu banyak dari node non terminal. Untuk setiap node non terminal diawali ID makanan, banyak childrennya, dan ID dari makanan yang menjadi childrennya. Kecuali root node, setiap non terminal yang didefinisikan harus telah didefinisikan sebelumnya (31 anak dari 37, 21 anak dari 31, dst).



```
src > konfigurasi > Peta.txt
1 10 10
2 #####
3 TK#BTC#X##
4 #M#F###X##
5 #####X##
6 ####XXXX##
7 #X#####
8 #X#####C#
9 #XXX##F###
10 #####
11 #####B###

src > konfigurasi > Resep.txt
1 2 You, 4 days ago | 2 authors (You and ot
2 3 You, 3 weeks ago •
3 37 3 31 15 15
4 31 2 21 10
5 21 1 11
6 2
7 7 2 5 6
8 6 1 77
```

```
src > konfigurasi > Makanan.t
1  10 You, 4 day
2  11
3  Ayam Mentah
4  1 0 0
5  0 0 15
6  Buy
7  21
8  Ayam Potong
9  0 2 0
10 0 0 0
11 Chop
```

3 Struktur Data (ADT)

3.1 ADT *Point*

- Sketsa struktur data: struktur data *point* berupa tipe data berbentuk tupel integer yang terdiri dari komponen X dan Y. ADT ini digunakan untuk merepresentasikan sebuah titik dari diagram kartesius
- Persoalan yang diselesaikan: untuk menampilkan posisi koordinat simulator berada pada peta.
- Alasan pemilihan: karena posisi simulator pada peta direpresentasikan sebagai matriks yang memiliki nilai absis (X) dan ordinat (Y) yang sesuai dengan ADT *Point*.
- Implementasi ADT *Point* terdapat pada file *point.h* dan *point.c*.

3.2 ADT *Time*

- Sketsa struktur data: struktur data yang merepresentasikan waktu, berupa tupel integer yang terdiri dari komponen hari, jam, dan menit.
- Persoalan yang diselesaikan: lama waktu proses pembelian, pengolahan, dan waktu kedaluwarsa bahan makanan.
- Alasan pemilihan: ADT *Time* memiliki struktur data yang tepat untuk dapat menyelesaikan beberapa persoalan terkait waktu dalam pemrosesan bahan makanan, karena memiliki validasi serta pembentukan struktur *Time*, membaca dan menuliskannya, konversi, serta beberapa operasi lainnya.
- Implementasi ADT *Time* terdapat pada file *time.h* dan *time.c*.

3.3 ADT Makanan

- Sketsa struktur data: struktur data yang berisi tupel yang berisi id, nama, aksi, masa kedaluwarsa, serta lama pengiriman dari bahan makanan yang tersedia, dan memiliki komponen integer lebar dan panjang.
- Persoalan yang diselesaikan: merepresentasikan bahan-bahan makanan yang tersedia pada simulator dan atribut-atributnya.
- Alasan pemilihan: untuk memudahkan penggunaan objek makanan dalam proses perpindahan maupun penghapusannya.

- Implementasi ADT Makanan terdapat pada file makanan.h dan makanan.c.

3.4 ADT Peta

- Sketsa struktur data: struktur data yang berupa tupel yang berisi lebar dan panjang peta yang direpresentasikan dalam tipe integer serta *pointer* untuk legenda yang tampil pada peta.
- Persoalan yang diselesaikan: untuk menggambarkan peta pada program simulator, di mana simulator dapat bergerak maupun lokasi bahan makanan dapat diproses.
- Alasan pemilihan: diperlukan representasi wilayah untuk simulator dapat berinteraksi mengolah bahan-bahan makanan, maupun wilayah di mana simulator tidak dapat melewatinya.
- Implementasi ADT Peta terdapat pada file peta.h dan peta.c.

3.5 ADT List Statik

- Sketsa struktur data: berupa array bertipe *alt-2a* yaitu eksplisit dan rata kiri dan memori list bersifat statis, yakni didefinisikan maksimal sebanyak 100.
- Persoalan yang diselesaikan: digunakan untuk membantu merepresentasikan ADT Makanan untuk ListMakanan serta ADT *Word*.
- Alasan pemilihan: diperlukan array yang bersifat statis untuk menyimpan beberapa elemen data.
- Implementasi ADT List Statik terdapat pada file liststatik.h dan liststatik.c.

3.6 ADT Matriks

- Sketsa struktur data: berupa array of array yang memiliki komponen baris dan kolom efektif yang direpresentasikan dalam integer, dan berukuran maksimum 100x100.
- Persoalan yang diselesaikan: digunakan dalam penentuan penempatan bahan makanan dalam kulkas serta pemrosesan representasi peta.
- Alasan pemilihan: memiliki komponen baris dan kolom, sehingga dapat menyelesaikan permasalahan yang memiliki representasi 2 dimensi.
- Implementasi ADT Matriks terdapat pada file matrix.h dan matrix.c.

3.7 ADT Mesin Karakter

- Sketsa struktur data: berupa representasi "mesin" yang mampu membaca sebuah pita berisi deret karakter untuk kemudian hasil pembacaan dapat dieksekusi. Memiliki komponen *mark* berupa '.' (titik) yang mengakhiri proses pembacaan pita karakter.
- Persoalan yang diselesaikan: digunakan dalam representasi pembacaan kata dalam mesin kata.
- Alasan pemilihan: merupakan ADT yang tepat dalam pembacaan input dari *command* program simulasi.
- Implementasi ADT Mesin Karakter terdapat pada file charmachine.h dan charmachine.c.

3.8 ADT Mesin Kata

- Sketsa struktur data: berupa representasi "mesin" yang mampu membaca input kalimat dan membaca kata per kata untuk kemudian hasil pembacaan dapat dieksekusi. Memiliki

STEI- ITB	IF2110-TB-G-2	Halaman 7 dari 36 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

komponen yang dapat memisahkan *blank* dan *mark* yang juga berupa '.' (titik) yang mengakhiri proses pembacaan kalimat.

- Persoalan yang diselesaikan: digunakan dalam membaca *command* dari program simulasi untuk kemudian dieksekusi secara tepat.
- Alasan pemilihan: ADT yang sesuai untuk pembacaan input karena program simulasi ini berupa CLI (*Command-line Interface*) di mana input dan output dilakukan di terminal.
- Implementasi ADT Mesin Kata terdapat pada file *wordmachine.h* dan *wordmachine.c*.

3.9 ADT Word

- Sketsa struktur data: berupa array penyimpan karakter yang membentuk sebuah kata, yang memiliki komponen integer panjang kata, untuk kemudian dapat dilakukan berbagai proses seperti membandingkan dua kata, mengubah tipe data dasar menjadi tipe data *word*, menyalin kata, dan lainnya.
- Persoalan yang diselesaikan: pembacaan input dari *command* dan beberapa konversi dari tipe data menjadi tipe *word* untuk kemudian mempermudah eksekusi.
- Alasan pemilihan: diperlukan sebuah tipe data yang memudahkan dalam pengolahan input, dan perbandingan serta pencocokan input dengan eksekusi yang akan dijalankan.
- Implementasi ADT *Word* terdapat pada file *word.h* dan *word.c*.

3.10 ADT Priority Queue Dinamis

- Sketsa struktur data: struktur data array yang statik, yang mengalokasikan memori tambahan ketika sudah penuh (ekspansi memori). Tersusun berurut berdasarkan data antrian yang diprioritaskan untuk satu hal tertentu untuk menjalankan konsep FIFO (First In First Out), di mana elemen akan di sort setiap kali terdapat proses enqueue. Menggunakan konsep array rata-kiri yang mengshift elemen ke kiri pada proses dequeue.
- Persoalan yang diselesaikan: penempatan makanan dengan prioritas waktu kedaluwarsa yang paling singkat, untuk dapat diolah terlebih dahulu agar tidak terbuang sia-sia.
- Alasan pemilihan: diperlukan sebuah tipe data yang mampu mengurutkan bahan makanan sesuai dengan waktu kedaluwarsa tersingkat agar dapat diolah lebih dahulu.
- Implementasi ADT *Priority Queue* Dinamis terdapat pada file *prioqueue Dinamik.h* dan *prioqueue Dinamik.c*.

3.11 ADT Stack

- Sketsa struktur data: struktur data dengan sederet elemen berupa tumpukan yang dikenali elemen puncaknya (*top*). Aturan yang digunakan yakni LIFO (Last In First Out), di mana *top* adalah satu-satunya lokasi terjadinya operasi, baik penghapusan maupun penambahan elemen.
- Persoalan yang diselesaikan: program dapat melakukan undo-redo, yakni membatalkan sebuah state tertentu (undo) atau kembali ke keadaan semula sebelum state tersebut dibatalkan (redo).
- Alasan pemilihan: diperlukan konsep ADT yang dapat mengubah (menambah/menghapus) suatu keadaan yang paling terbaru terjadi (Last In First Out) dalam konsep undo-redo.
- Implementasi ADT *Stack* terdapat pada file *stack.h* dan *stack.c*.

3.12 ADT List Linier

- Sketsa struktur data: struktur data list yang berupa *node* yang terkait dengan *node* lain. *Node* merupakan sebuah tupel yang terdiri dari sebuah nilai tertentu (info) dan penunjuk ke *node* lainnya (next). Hal ini memungkinkan penyimpanan elemen-elemen tanpa harus kontigu.
- Persoalan yang diselesaikan: digunakan dalam membantu pembentukan ADT *Tree* dan ADT Peta.
- Alasan pemilihan: diperlukan konsep ADT yang dapat mengakses elemen tanpa harus diurutkan sesuai indeksinya seperti pada list statik biasa, namun tinggal menunjuk alamat (*address*) *node* selanjutnya.
- Implementasi ADT List Linier terdapat pada file *listlinier.h* dan *listlinier.c*.

3.13 ADT Set

- Sketsa struktur data: berupa kumpulan objek-objek yang unik dan tidak memerdulikan urutannya. Set yang digunakan berisi kumpulan id makanan yang tersedia pada program simulasi. Indeks pada set merupakan urutan kemunculan makanan di file config.
- Persoalan yang diselesaikan: digunakan untuk penentuan rekomendasi makanan yang dapat dibuat dari bahan-bahan makanan yang dimiliki.
- Alasan pemilihan: rekomendasi makanan dapat menggunakan konsep subset untuk bisa menentukan apakah suatu resep makanan dapat dibuat, jika resep yang diperlukan merupakan subset dari bahan makanan yang dimiliki.
- Implementasi ADT Set terdapat pada file *set.h* dan *set.c*.

3.14 ADT Tree

- Sketsa struktur data: berupa array of pohon N-ary dengan representasi pointer. Memiliki komponen elemen simpul utama (*root*), memiliki *children* yakni simpul yang diturunkan dari suatu simpul, dan memiliki subpohon sebanyak maksimum N. Dapat menyimpan banyak root dari tree yang tidak berhubungan pada indeks array yang berbeda.
- Persoalan yang diselesaikan: digunakan untuk menyimpan resep (penurunan resep (*node*) dari bahan-bahan yang diperlukan (*child*)).
- Alasan pemilihan: penggunaannya sangat cocok karena mampu merepresentasikan proses menggabungkan 2 atau lebih bahan makanan menjadi satu (dari beberapa *child* menjadi satu *node*).
- Implementasi ADT *Tree* terdapat pada file *tree.h* dan *tree.c*.

3.15 ADT Kulkas

- Sketsa struktur data: berupa representasi matriks yang dapat menyimpan bahan makanan sesuai dengan panjang dan lebar bahan makanan tersebut, serta memiliki komponen jumlah makanan bertipe integer.
- Persoalan yang diselesaikan: digunakan untuk menyimpan bahan makanan dari *inventory* ke kulkas agar *inventory* tidak terlalu penuh.
- Alasan pemilihan: digunakan konsep matriks karena setiap bahan makanan memiliki panjang dan lebar tertentu sehingga dapat dikira-kira penempatannya pada kulkas tersebut (tidak bisa menumpuk).

- Implementasi ADT Kulkas terdapat pada file *kulkas.h* dan *kulkas.c*.

4 Program Utama

Program utama yang dijalankan bernama *main.exe*, hasil *compile* dari file *main.c* yang telah meng-include semua ADT dan file yang berisi *command*, *simulator*, dan *configParser*. Setelah program dijalankan, akan memunculkan *splash art* yang menandakan program telah dimulai. Pengguna diminta untuk menginput namanya. Setelah itu, pengguna dapat memilih untuk START atau EXIT program. Jika pengguna salah memasukkan input, program akan *looping* sampai pengguna memasukkan input yang tepat. Jika pengguna memilih EXIT, maka program akan langsung berhenti. Jika pengguna memilih START, maka program dijalankan, dan akan muncul info program yang terdiri dari nama pengguna, waktu yang berjalan, notifikasi, serta peta yang menunjukkan di mana simulator berada dan berbagai *station* yang dapat melakukan berbagai hal. Selain itu juga, pengguna diminta lagi untuk menginput *command* untuk melanjutkan program pada *command prompt* yang tersedia. Jika input tidak sesuai dengan fitur yang ada, program akan menolak dan mengirimkan pemberitahuan serta melakukan *looping* sampai pengguna menginput perintah yang sesuai dengan fitur yang tersedia.

Untuk pertama kali, pengguna disarankan untuk menginput *command HELP* untuk mengetahui fitur-fitur apa saja yang tersedia dan dapat dijalankan. Pengguna dapat berpindah tempat dengan menginput *command MOVE* sesuai dengan jalan pada peta yang tersedia. Setiap kali pengguna berpindah tempat, waktu akan bertambah satu menit. Pengguna tidak bisa menabrak dinding (X), pembatas peta (*), maupun *station* yang tersedia. Pengguna akan diperingati jika bergerak menabrak benda tersebut. *Station-station* yang tersedia dapat digunakan dengan *command* yang sesuai jika simulator (S) berada *adjacent* (tidak berlaku diagonal) dengan *station* yang diinginkan.

Pengguna dapat membeli bahan makanan pada *station T*. Makanan yang dibeli memiliki waktu pengiriman tertentu. Pengguna dapat mengecek status makanan dalam pengiriman dengan *command DELIVERY*. Jika makanan telah sampai, akan muncul notifikasi bahwa makanan telah sampai. Pengguna kemudian dapat mengolah makanan yang sudah dibeli. Makanan dapat di-*chop* di C, dapat di-*fry* di F, di-*mix* di M, serta di-*boil* di B. Jika pengguna tidak memiliki bahan makanan, maka akan muncul peringatan bahwa tidak ada bahan makanan yang dapat diolah. Pengolahan bahan makanan juga memerlukan waktu tertentu.

Pengguna dapat mengecek daftar makanan yang tersedia dengan *command CATALOG*, dapat mengecek resep dengan *command COOKBOOK*, mengecek *inventory* dari bahan-bahan yang telah dimiliki dengan *command INVENTORY*, serta mengecek rekomendasi resep makanan yang sesuai dengan bahan makanan yang dimiliki di *inventory* dengan *command REKOMENDASI*. Pengguna juga dapat menjalankan *command WAIT X Y*, dengan X adalah lamanya jam dan Y adalah lamanya menit untuk maju ke durasi waktu tertentu tanpa harus memindahkan simulator.

Makanan memiliki waktu kedaluwarsanya masing-masing, jika suatu saat makanan telah melewati masa kedaluwarsa, maka akan muncul notifikasi yang memberitahu suatu makanan telah kedaluwarsa dan tidak bisa digunakan. Untuk mengantisipasi masa kedaluwarsa, bahan makanan dapat disimpan ke dalam kulkas yang berukuran 10x20 (dapat dikonfigurasi). Untuk memasukkan makanan ke dalam kulkas simulator perlu berada *adjacent* di *station K* dan menjalankan *command KULKAS*. Jika *inventory* kosong, maka akan muncul pemberitahuan tidak ada bahan makanan

STEI- ITB	IF2110-TB-G-2	Halaman 10 dari 36 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

yang dapat disimpan di kulkas. Selain itu, jika kulkas penuh, muncul juga pemberitahuan tidak dapat menyimpan apapun lagi di kulkas. Jika kulkas kosong dan pengguna hendak mengambil makanan dari kulkas juga akan memunculkan peringatan. Isi kulkas dapat di-*display* untuk dilihat ketersediaan ruang dan ID makanan yang tersimpan.

Jika pengguna melakukan sebuah kesalahan, pengguna dapat menjalankan *command UNDO* ataupun *REDO*. Pengguna dapat *undo* jika ingin kembali ke *state* sebelumnya. Jika pengguna belum melakukan apapun, maka akan muncul peringatan untuk gagal *undo* karena belum ada *state* terbentuk. Pengguna dapat *redo* jika ingin maju ke *state* sebelum di-undo. Jika pengguna belum meng-*undo* apapun atau sedang berada di *state* paling akhir, akan muncul peringatan untuk gagal *redo*.

Setelah selesai, pengguna dapat menyelesaikan program dan keluar dari program dengan *command EXIT*. Program akan selesai dan sayangnya tidak disimpan ke file manapun. Pengguna dapat merubah program dari awal dengan mengubah konfigurasi file yang ingin digunakan.

5 Algoritma-Algoritma Menarik

5.1 Algoritma Undo-Redo

Algoritma pada fitur Undo-Redo yang dilakukan dengan konsep *pop* dan *push stack* dari sebuah *state*. Algoritma ini tergolong menarik karena kita dapat membuat pengguna menjalankan program sesuai dengan kehendaknya. Dengan artian, ketika pengguna melakukan kesalahan, pengguna bisa kembali dan memperbaiki untuk menghindari kesalahan tersebut secara tak terbatas memori.

Implementasi Undo-Redo juga perlu diperhatikan karena hanya *command* yang melakukan sesuatu yang bisa di-undo dan tidak semua *command* akan disimpan ke dalam *stack undo* sedangkan untuk *stack redo* merupakan isi dari *stack undo* yang sudah di-*pop*, tetapi jika user melakukan *command* yang lain maka *stack redo* tersebut akan dihapus.

5.2 Algoritma Set-Tree Rekomendasi Makanan

Algoritma yang memadukan konsep *set* dan *tree* untuk menentukan rekomendasi resep makanan ini juga tergolong menarik karena cukup memiliki berbagai alternatif. Karena resep yang digunakan di sini disimpan sebagai *tree*, namun diperlukan konsep subset untuk menentukan rekomendasi makanan yang sesuai dengan *inventory* yang dimiliki. Terdapat beberapa konsep algoritma yang juga dapat digunakan yakni salah satunya *union-find algorithm*.

Ketika ada elemen yang dibutuhkan resep tetapi tidak ada di tas, akan dilihat status dari elemen tersebut. Jika elemen tersebut terminal (dari hasil BUY), maka dapat disimpulkan bahwa resep tidak bisa dibuat. Akan tetapi jika resep tersebut bukan terminal, maka resep tersebut akan dicoba untuk dicari elemen pembuatannya, dan di union dengan resep awalnya yang tidak lagi mengandung elemen tersebut. Proses ini diulangi dan jika segala syarat terpenuhi, maka resep dikatakan dapat dibuat.

STEI- ITB	<i>IF2110-TB-G-2</i>	Halaman 12 dari 36 halaman
<p>Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB</p>		

```

Hello? Siapa disitu? (tanpa spasi) : Iwan
Hi Iwan!
Silahkan Pilih Command Berikut:
START
EXIT
Command Prompt: START
Game started
Loading...
Load success!

Player Name: Iwan
Time: 0
Notifikasi:

Map:
* * * * *
* S *
* T K B T C X *
* M F X *
* X *
* X X X X *
* X *
* X C *
* X X X F *
* *
* B *
* * * * *
Command Prompt: █

```

Gambar 6.2. Tampilan *Command START*

```

Hello? Siapa disitu? (tanpa spasi) : Iwan
Hi Iwan!
Silahkan Pilih Command Berikut:
START
EXIT
Command Prompt: EXIT
Game exited

```

Gambar 6.3. Tampilan *Command EXIT*

6.3 Help

Jika pengguna memasukkan *command HELP*, akan ditampilkan berbagai *command* yang dapat dijalankan sesuai dengan keinginan pengguna.

```
Command Prompt: HELP

-----
Daftar Command Tersedia
-----
Command Program
-----
1. CATALOG - Melihat daftar makanan tersedia
2. COOKBOOK - Melihat daftar resep tersedia
3. WAIT (X) (Y) - Menunggu selama X jam dan Y menit
4. MOVE NORTH/EAST/SOUTH/WEST - Bergerak satu langkah ke arah yang dituju
5. UNDO - Mengulang ke keadaan sebelumnya
6. REDO - Mengulangi apa yang sudah di UNDO
7. INVENTORY - Melihat isi inventory
7. REKOMENDASI - Melihat rekomendasi masak
-----
```

Gambar 6.4. Tampilan *Command HELP*

6.4 Catalog

Jika pengguna meng-input *command CATALOG*, akan ditampilkan daftar makanan yang tersedia yang diikuti oleh data waktu kedaluwarsanya, aksi yang diperlukan, dan lama waktu pemrosesannya.

```
Command Prompt: CATALOG
1. Ayam Mentah - 1 hari - BUY - 15 menit
2. Ayam Potong - 2 jam - CHOP - 0
3. Tepung - 2 jam - BUY - 30 menit
4. Minyak Goreng - 2 jam 30 menit - BUY - 1 jam
5. Ayam Tepung - 2 jam 30 menit - MIX - 2 menit
6. Ayam Goreng - 2 jam 15 menit - FRY - 5 menit
7. Spaghetti Bolognaise - 2 jam 15 menit - MIX - 5 menit
8. Spaghetti - 2 jam 15 menit - BUY - 5 menit
9. Bolognaise - 2 jam 15 menit - MIX - 5 menit
10. Tomato - 2 jam 15 menit - BUY - 5 menit
```

Gambar 6.5. Tampilan *Command CATALOG*

6.5 Cookbook

Jika pengguna meng-input *command COOKBOOK*, akan ditampilkan daftar resep makanan yang tersedia beserta bahan-bahan makanan yang dibutuhkan untuk membuatnya.

```

Command Prompt: COOKBOOK
Hai! Ini subbagian dari resep ke - 0
ID Makanan: 7
Spaghetti Bolognaise
----MIX----
Bahan yang dibutuhkan:
Food ID: 6 ---Bolognaise---
Food ID: 5 ---Spaghetti---

ID Makanan: 6
Bolognaise
----MIX----
Bahan yang dibutuhkan:
Food ID: 77 ---Tomato---

ID Makanan: 77
Tomato
----BUY----
Beli di olshop pake telefon ya, ribet nyarinya.

ID Makanan: 5
Spaghetti
----BUY----
Beli di olshop pake telefon ya, ribet nyarinya.

-----
Hai! Ini subbagian dari resep ke - 1
ID Makanan: 37
Ayam Goreng
----FRY----
Bahan yang dibutuhkan:
Food ID: 15 ---Minyak Goreng---
Food ID: 15 ---Minyak Goreng---
Food ID: 31 ---Ayam Tepung---

ID Makanan: 15
Minyak Goreng
----BUY----
Beli di olshop pake telefon ya, ribet nyarinya.

ID Makanan: 31
Ayam Tepung
----MIX----
Bahan yang dibutuhkan:
Food ID: 10 ---Tepung---
Food ID: 21 ---Ayam Potong---

```

Gambar 6.6. Tampilan *Command COOKBOOK*

6.6 Move

Pengguna dapat meng-input *command MOVE NORTH/EAST/SOUTH/WEST* untuk menggerakkan simulator pada peta. Setiap kali simulator bergerak, maka *Time* akan bertambah sebanyak 1 menit.

STEI- ITB	IF2110-TB-G-2	Halaman 15 dari 36 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

```

Map:
* * * * *
* S           *
* T K   B T C   X   *
*   M   F       X   *
*               X   *
*           X X X X   *
*   X           *
*   X           C   *
*   X X X       F   *
*               *
*               B   *
* * * * *

Command Prompt: MOVE EAST

Player Name: Iwan
Time: 1 menit
Notifikasi:

Map:
* * * * *
* S           *
* T K   B T C   X   *
*   M   F       X   *
*               X   *
*           X X X X   *
*   X           *
*   X           C   *
*   X X X       F   *
*               *
*               B   *
* * * * *

```

Gambar 6.7. Tampilan *Command MOVE*

Selain itu, jika simulator bergerak menabrak pembatas peta, akan muncul notifikasi “*out of bound*”, dan jika menabrak dinding (X) atau lokasi lainnya (T, K, M, B, C, F) akan muncul notifikasi “*collision*” dan Simulator tidak bergerak ke manapun.


```

Map:
* * * * *
*      S      *
* T K   B T C   X   *
*   M   F       X   *
*               X   *
*           X X X X   *
*   X           *
*   X           C   *
*   X X X       F   *
*               *
*               B   *
* * * * *

Command Prompt: MOVE NORTH

Player Name: Iwan
Time: 2 menit
Notifikasi:
1. Out of bound

```

Gambar 6.8. Tampilan Ketika Simulator Menabrak Pembatas Peta

<pre> Map: * * * * * * S * * T K S B T C X * * M F X * * X * * X X X X * * X * * X C * * X X X F * * * * B * * * * * * Command Prompt: MOVE EAST Player Name: Iwan Time: 3 menit Notifikasi: 1. Collision </pre>	<pre> Map: * * * * * * S * * T K B T C X * * M F X * * X * * X X X X * * X * * X S C * * X X X F * * * * B * * * * * * Command Prompt: MOVE SOUTH Player Name: Iwan Time: 8 menit Notifikasi: 1. Collision </pre>
--	---

Gambar 6.9. Tampilan Ketika Simulator Menabrak lokasi *Boil* (B) dan Dinding (X)

6.7 Wait

Pengguna dapat meng-input *command WAIT* untuk langsung memperoleh durasi waktu yang diinginkan tanpa harus menggerakkan simulator. Durasi waktu yang diinput akan langsung dijumlahkan dengan waktu sekarang.

```

Player Name: Iwan
Time: 8 menit
Notifikasi:

Map:
* * * * *
*
* T K   B T C   X   *
*   M   F       X   *
*               X   *
*           X X X X   *
*   X               *
*   X S               C *
*   X X X           F   *
*                   *
*               B       *
* * * * *
Command Prompt: WAIT 1 10

Player Name: Iwan
Time: 1 jam 18 menit
Notifikasi:

```

Gambar 6.10. Tampilan *Command WAIT* Selama 1 Jam 10 Menit

6.8 Buy, Delivery, Boil, Fry, Chop, dan Mix

Pengguna dapat melakukan berbagai hal untuk memperoleh dan mengolah suatu makanan. Pengguna dapat membeli makanan (*BUY*) di *station T*, merebus (*BOIL*) dapat dilakukan di *station B*, menggoreng (*FRY*) dapat dilakukan di *station F*, memotong (*CHOP*) dapat dilakukan di *station C*, dan mencampur (*MIX*) dapat dilakukan di *station M*.

```

Command Prompt: BUY
=====BUY=====
ID (11). Ayam Mentah --- Duration: 15 menit
ID (10). Tepung --- Duration: 30 menit
ID (15). Minyak Goreng --- Duration: 1 jam
ID (5). Spaghetti --- Duration: 5 menit
ID (77). Tomato --- Duration: 5 menit
Masukkan ID MAKANAN yang anda inginkan!
Jika berubah pikiran/tidak ada yang memenuhi, masukkan 0

Jadi, mau pilih apa? : 11

Player Name: Iwan
Time: 1 jam 26 menit
Notifikasi:
    1. Baik, nanti kami kirim... Item: Ayam Mentah --- Lama Masuk Tas: 15 menit

```

Gambar 6.11. Tampilan Membeli Bahan Makanan

Pengguna dapat mengetahui bahan makanan yang sedang diantar dengan *command DELIVERY*.

STEI- ITB	IF2110-TB-G-2	Halaman 18 dari 36 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

```

Command Prompt: delivery
=====DELIVERY=====
1. Ayam Mentah ---- Sisa waktu sebelum sampai: 15 menit
2. Minyak Goreng ---- Sisa waktu sebelum sampai: 1 jam

```

Gambar 6.12. Tampilan Bahan Makanan yang Sedang diantar

Ketika waktu *delivery* makanan telah terpenuhi, akan muncul notifikasi bahwa makanan tersebut telah sampai.

```

Player Name: Iwan
Time: 2 jam 30 menit
Notifikasi:
1. Hore! Makanan ini telah sampai: Minyak Goreng
2. Hore! Makanan ini telah sampai: Tepung
3. Hore! Makanan ini telah sampai: Tepung
4. Hore! Makanan ini telah sampai: Ayam Mentah
5. Hore! Makanan ini telah sampai: Spaghetti
6. Hore! Makanan ini telah sampai: Tomato

```

Gambar 6.13. Tampilan Notifikasi Makanan yang Dibeli Telah Sampai

Pengguna dapat memotong bahan makanan dengan *command CHOP* untuk memperoleh bahan makanan tertentu.

```

Command Prompt: CHOP
=====CHOP=====
ID (21). Ayam Potong --- Duration: 0
Masukkan ID MAKANAN yang anda inginkan!
Jika berubah pikiran/tidak ada yang memenuhi, masukkan 0

Jadi, mau pilih apa? : 21

Player Name: Iwan
Time: 2 jam 39 menit
Notifikasi:
1. Yay! Makanan ini berhasil dibuat: Ayam Potong

```

Gambar 6.14. Tampilan Ketika Berhasil Memotong Ayam Mentah

Jika bahan makanan yang ingin dipotong tidak ada, maka akan muncul notifikasi seperti gambar berikut.

```

Command Prompt: CHOP
=====CHOP=====
ID (21). Ayam Potong --- Duration: 0
Masukkan ID MAKANAN yang anda inginkan!
Jika berubah pikiran/tidak ada yang memenuhi, masukkan 0

Jadi, mau pilih apa? : 21
Yahh, kamu gak bisa bikin Ayam Potong karena kamu gak punya:
1. Ayam Mentah

```

Gambar 6.15. Tampilan *CHOP* Bahan Makanan

Pengguna dapat merebus bahan makanan yang dapat direbus. Namun, jika tidak ada bahan makanan yang bisa direbus tidak akan muncul makanan apapun.

```

Command Prompt: BOIL
=====BOIL=====
Masukkan ID MAKANAN yang anda inginkan!
Jika berubah pikiran/tidak ada yang memenuhi, masukkan 0

Jadi, mau pilih apa? : 0
Tolong jangan ngerjain saya ya.. Banyak tubes tau cape..

```

Gambar 6.16. Tampilan *BOIL* Bahan Makanan

Selanjutnya, pengguna dapat mencampurkan bahan makanan dengan bahan makanan lainnya, jika telah selesai, akan muncul notifikasi bahwa makanan telah selesai di-*mix*.

```

Command Prompt: MIX
=====MIX=====
ID (31). Ayam Tepung --- Duration: 2 menit
ID (7). Spaghetti Bolognese --- Duration: 5 menit
ID (6). Bolognese --- Duration: 5 menit
Masukkan ID MAKANAN yang anda inginkan!
Jika berubah pikiran/tidak ada yang memenuhi, masukkan 0

Jadi, mau pilih apa? : 31

Player Name: Iwan
Time: 2 jam 44 menit
Notifikasi:
    1. Krrsss... Ayam Tepung akan selesai dibuat dalam 2 menit

```

Gambar 6.17. Tampilan Setelah Mencampurkan Ayam dan Tepung

```

Player Name: Iwan
Time: 2 jam 46 menit
Notifikasi:
    1. Yay! Makanan ini berhasil dibuat: Ayam Tepung

```

Gambar 6.18. Notifikasi Ayam dan Tepung Selesai di-*Mix*

Jika ada bahan yang tidak dimiliki, maka akan gagal melakukan *mix* dan muncul pemberitahuan bahan yang kurang.

```
Command Prompt: MIX
=====MIX=====
ID (31). Ayam Tepung --- Duration: 2 menit
ID (7). Spaghetti Bolognaise --- Duration: 5 menit
ID (6). Bolognaise --- Duration: 5 menit
Masukkan ID MAKANAN yang anda inginkan!
Jika berubah pikiran/tidak ada yang memenuhi, masukkan 0

Jadi, mau pilih apa? : 31
Yahh, kamu gak bisa bikin Ayam Tepung karena kamu gak punya:
1. Ayam Potong
```

Gambar 6.19. Tampilan Gagal *Mix* Bahan Makanan

Sama seperti *command* sebelumnya, ketika *FRY* dijalankan pengguna dapat menggoreng suatu bahan makanan. Dan ketika selesai digoreng akan muncul notifikasi.

```
=====FRY=====
ID (37). Ayam Goreng --- Duration: 5 menit
Masukkan ID MAKANAN yang anda inginkan!
Jika berubah pikiran/tidak ada yang memenuhi, masukkan 0

Jadi, mau pilih apa? : 37

Player Name: Iwan
Time: 2 jam 46 menit
Notifikasi:
1. Krrsss... Ayam Goreng akan selesai dibuat dalam 5 menit
```

Gambar 6.20. Tampilan Menggoreng Makanan

```
Player Name: Iwan
Time: 2 jam 51 menit
Notifikasi:
1. Yay! Makanan ini berhasil dibuat: Ayam Goreng
```

Gambar 6.21. Notifikasi Ayam Telah selesai Digoreng

Ketika tidak memiliki bahan makanan yang diperlukan untuk menggoreng, juga muncul pemberitahuan.

```

Command Prompt: FRY
=====FRY=====
ID (37). Ayam Goreng --- Duration: 5 menit
Masukkan ID MAKANAN yang anda inginkan!
Jika berubah pikiran/tidak ada yang memenuhi, masukkan 0

Jadi, mau pilih apa? : 37
Yahh, kamu gak bisa bikin Ayam Goreng karena kamu gak punya:
    1. Ayam Tepung
    2. Minyak Goreng

```

Gambar 6.22. Tampilan Gagal Menggoreng Makanan

Simulator harus berada *adjacent* (tidak berlaku diagonal) untuk bisa melakukan *command-command* tersebut. Jika tidak berada di lokasi yang seharusnya, akan muncul pemberitahuan.

```

Map:
* * * * *
*           *
* T K S B T C   X   *
*   M   F       X   *
*           X       *
*       X X X X     *
*   X           *
*   X           C   *
*   X X X       F   *
*           B       *
* * * * *

Command Prompt: MIX
Maaf, anda tidak berada di station yang tepat untuk melakukan perintah.
Tolong jangan ngerjain saya ya.. Banyak tubes tau cape..

```

Gambar 6.23. Tampilan Gagal Melakukan *Command* Karena Tidak *Adjacent* Dengan *station* yang tepat

6.9 Inventory

Pengguna dapat mengetahui bahan-bahan yang dimilikinya dan telah tersimpan di *inventory*. Akan muncul daftar makanan beserta waktu kedaluwarsanya.

```

Command Prompt: inventory
=====INVENTORY=====
    1. Tomato ---- Sisa waktu sebelum busuk: 58 menit
    2. Tepung ---- Sisa waktu sebelum busuk: 1 jam 8 menit
    3. Ayam Goreng ---- Sisa waktu sebelum busuk: 2 jam 14 menit

```

Gambar 6.24. Tampilan *Inventory*

6.10 Kulkas

Pengguna dapat memindahkan bahan makanan dari *Inventory* dan disimpan ke dalam kulkas untuk menghindari masa kedaluwarsa. Pengguna harus *adjacent* di *station K*.

```
Command Prompt: kulkas
=====INVENTORY=====
1. Tomato ---- Sisa waktu sebelum busuk: 58 menit
2. Tepung ---- Sisa waktu sebelum busuk: 1 jam 8 menit
3. Ayam Goreng ---- Sisa waktu sebelum busuk: 2 jam 14 menit

===== MENU KULKAS =====
Note: Pilihan selain 1 atau 2 akan diprint isi dari kulkas!
      Selain itu, setiap kali kamu masukan makanan ke kulkas,
      Makanan akan langsung disimpan secara teratur oleh mesin!
      Jadi, kamu gak harus cape-cape lagi nyari tempat nyimpen makanan.
=====
Pilihan untuk menu kulkas:
1. (KULKAS) SIMPAN (ID INV)
2. (KULKAS) AMBIL (ID ITEM KULKAS)
3. (KULKAS) DISPLAY
=====
Ketik pilihan (1/2/3): █
```

Gambar 6.25. Tampilan Ketika *Command* kulkas dijalankan

Pengguna dapat menentukan apa yang akan dilakukan di kulkas, terdapat menu simpan ke kulkas, ambil dari kulkas, serta menampilkan isi kulkas (*display*).

```
Pilihan untuk menu kulkas:
1. (KULKAS) SIMPAN (ID INV)
2. (KULKAS) AMBIL (ID ITEM KULKAS)
3. (KULKAS) DISPLAY
=====
Ketik pilihan (1/2/3): 1
Jadi, mau masukan item nomor berapa di kulkas? : 1
Masukkan lebar dan panjang item mu (lebar) (panjang): 2 4
Tomato (77) berhasil masuk ke dalam kulkas!
Registered ID makanan dalam Kulkas: 1
sukses
```

Gambar 6.26. Tampilan Berhasil Menyimpan Makanan ke Kulkas

Terlihat dibutuhkan data panjang dan lebar bahan makanan untuk menyesuaikan kapasitas kulkas. Jika *inventory* kosong, muncul pemberitahuan bahwa tidak ada bahan makanan yang dapat disimpan.

```
Command Prompt: kulkas
=====INVENTORY=====
Kosong :(

===== MENU KULKAS =====
Note: Pilihan selain 1 atau 2 akan diprint isi dari kulkas!
      Selain itu, setiap kali kamu masukin makanan ke kulkas,
      Makanan akan langsung disimpan secara teratur oleh mesin!
      Jadi, kamu gak harus cape-cape lagi nyari tempat nyimpen makanan.
=====
Pilihan untuk menu kulkas:
1. (KULKAS) SIMPAN (ID INV)
2. (KULKAS) AMBIL (ID ITEM KULKAS)
3. (KULKAS) DISPLAY
=====
Ketik pilihan (1/2/3): 1
Inventorymu kosong!
```

Gambar 6.27. *Inventory* Kosong, Tidak Dapat Menyimpan Apapun ke Kulkas

Jika kulkas telah penuh, akan muncul juga pemberitahuan bahwa kulkas tidak dapat menampung bahan makanan lagi.

```
=====
Pilihan untuk menu kulkas:
1. (KULKAS) SIMPAN (ID INV)
2. (KULKAS) AMBIL (ID ITEM KULKAS)
3. (KULKAS) DISPLAY
=====
Ketik pilihan (1/2/3): 1
Jadi, mau masukin item nomor berapa di kulkas? : 2
Masukkan lebar dan panjang item mu (lebar) (panjang): 1 2
Maaf, tidak cukup ruang untuk memasukkan makanan ini!
```

Gambar 6.28. Kulkas Penuh

Selain itu, untuk menampilkan isi kulkas dapat memilih input 3.

```
Pilihan untuk menu kulkas:
1. (KULKAS) SIMPAN (ID INV)
2. (KULKAS) AMBIL (ID ITEM KULKAS)
3. (KULKAS) DISPLAY
=====
Ketik pilihan (1/2/3): 3
===== ISI KULKAS =====
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
===== INFO =====
(ID MAKANAN KULKAS: 1) => NAMA: Tomato => DURASI EXPIRED: 58 menit
```

Gambar 6.29. Tampilan *Display* Isi Kulkas

Jika pengguna ingin mengambil bahan makanan dari kulkas dan memasukkannya ke *inventory*, dapat memilih input 2.

```
Ketik pilihan (1/2/3): 2
===== ISI KULKAS =====
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
===== INFO =====
(ID MAKANAN KULKAS: 1) => NAMA: Tepung => DURASI EXPIRED: 29 menit

Jadi, mau ambil item nomor berapa di kulkas? : 1

Player Name: Iwan
Time: 2 jam 1 menit
Notifikasi:
    1. Berhasil mengeluarkan makanan dari kulkas!
```

Gambar 6.30. Tampilan Mengambil Bahan Makanan dari Kulkas

Jika kulkas kosong, maka tidak ada makanan apapun yang dapat diambil dan akan muncul pemberitahuan. _____

```
Ketik pilihan (1/2/3): 2
===== ISI KULKAS =====
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
===== INFO =====

Jadi, mau ambil item nomor berapa di kulkas? : 1
ID makanan di kulkas tidak valid!
```

Gambar 6.31. Gagal Mengambil karena Kulkas Kosong

6.11 Rekomendasi Makanan

Pengguna dapat mengetahui rekomendasi resep makanan yang dapat dibuat berdasarkan bahan makanan yang dimilikinya dan cara mengolahnya dengan meng-input *command* REKOMENDASI.

```
Command Prompt: rekomendasi
```

```
List Resep yang Dapat Dibuat:
```

```
1. Ayam Potong---CHOP
```

```
2. Ayam Tepung---MIX
```

```
3. Ayam Goreng---FRY
```

```
4. Spaghetti Bolognaise---MIX
```

```
5. Bolognaise---MIX
```

Gambar 6.32. Daftar Rekomendasi Makanan Beserta Cara Mengolahnnya

Jika *inventory* kosong, maka tidak ada resep makanan yang direkomendasikan.

```
Command Prompt: REKOMENDASI
List Resep yang Dapat Dibuat:
Tidak ada resep yang dapat dibuat.
```

Gambar 6.33. Tidak Ada Rekomendasi saat *Inventory* Kosong

6.12 Makanan Kedaluwarsa

Ketika suatu bahan makanan telah melewati masa kedaluwarsanya, akan muncul notifikasi bahwa makanan tersebut telah kedaluwarsa.

```
Player Name: Iwan
Time: 11 jam 2 menit
Notifikasi:
  1. Makanan ini telah expired: Tepung
  2. Makanan ini telah expired: Spaghetti
```

Gambar 6.34. Tampilan Notifikasi Makanan Kedaluwarsa

6.13 Undo

Jika pengguna melakukan kesalahan atau hendak kembali ke *state* sebelumnya, pengguna dapat meng-input *command* UNDO untuk Kembali satu *state* sebelumnya.

```
Command Prompt: undo
=====UNDO=====
Undo berhasil

Player Name: Iwan
Time: 2 jam 1 menit
Notifikasi:
  1. Berhasil mengeluarkan makanan dari kulkas!

Map:
* * * * *
*   S   *
* T K   B T C   X   *
*   M   F       X   *
*           X   *
*       X X X X   *
*   X           *
*   X           C   *
*   X X X       F   *
*           *
*           B   *
* * * * *
Command Prompt: undo
=====UNDO=====
Undo berhasil

Player Name: Iwan
Time: 2 jam 1 menit
Notifikasi:
  1. Makanan dimasukkan ke kulkas dari kulkas
```

Gambar 6.35. UNDO Sebanyak 2 Kali

Jika belum melakukan apapun, maka gagal untuk undo.

```
Command Prompt: undo
=====UNDO=====
Tidak ada state yang bisa di undo
```

Gambar 6.36. Gagal Undo

6.14 Redo

Berkebalikan dengan UNDO, *command* REDO dapat digunakan ketika pengguna berubah pikiran setelah meng-undo sesuatu untuk kembali ke *state* sebelum di-undo.

```
Command Prompt: redo
=====REDO=====
Redo berhasil

Player Name: Iwan
Time: 2 jam 1 menit
Notifikasi:
  1. Berhasil mengeluarkan makanan dari kulkas!
```

Gambar 6.37. Notifikasi Setelah Melakukan REDO

Jika pengguna berada di *state* terbaru, artinya tidak pernah meng-undo sesuatu, akan muncul pemberitahuan.

```
Command Prompt: redo
=====REDO=====
Tidak ada state yang bisa di redo
```

Gambar 6.38. Gagal REDO

7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	<i>Compile</i> Program	Mengetahui apakah program dapat di- <i>compile</i> dan dapat dijalankan.	Melakukan perintah <i>compile</i> seperti pada perintah di atas pada terminal, lalu mencoba menjalankan file hasil <i>compiling</i> .	Data Test 6.1	Program dapat di- <i>compile</i> dan program utama dapat dijalankan.	Program telah ter- <i>compile</i> dan program utama dapat dijalankan.
2	Start dan Exit Program	Mengetahui apakah <i>command start</i> dan <i>exit</i> dapat berfungsi.	Menjalankan perintah START dan EXIT pada program berjalan.	Data Test 6.2	Program dapat dimulai dan diakhiri dengan baik.	Program telah dimulai dan diakhiri dengan baik sesuai <i>command</i> .
3	Help	Mengetahui apakah fitur-fitur yang dapat diakses pengguna dengan menjalankan <i>command</i> tertentu dapat ditampilkan.	Menjalankan <i>command</i> HELP.	Data Test 6.3	Program dapat menampilkan berbagai <i>command</i> fitur yang tersedia berdasarkan deskripsi fiturnya.	Program telah menampilkan berbagai <i>command</i> fitur yang tersedia berdasarkan deskripsi fiturnya.

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
4	Catalog	Mengetahui apakah daftar makanan yang tersedia, lama waktu mendapatkannya, aksi yang diperlukan, serta lama waktu kedaluwarsanya dapat ditampilkan.	Menjalankan <i>command</i> CATALOG	Data Test 6.4	Program dapat menampilkan daftar makanan yang tersedia berdasarkan beberapa data pendukungnya.	Program telah menampilkan daftar makanan yang tersedia berdasarkan beberapa data pendukungnya.
5	CookBook	Mengetahui apakah resep makanan beserta bahan-bahan yang diperlukan dapat ditampilkan.	Menjalankan <i>command</i> COOKBOOK	Data Test 6.5	Program dapat menampilkan daftar resep tersedia.	Program telah menampilkan daftar resep tersedia.
6	Move	Mengetahui apakah simulator dapat bergerak pada peta sesuai dengan aturan bergerak dan tidak menabrak.	Menjalankan <i>command</i> MOVE baik NORTH, EAST, SOUTH, dan WEST, serta sengaja menabrakkan simulator ke dinding dan <i>station</i> serta ke pembatas peta.	Data Test 6.6	Simulator dapat dijalankan sesuai dengan aturan dan tidak menabrak pembatas dinding/ <i>station</i> .	Simulator telah berjalan sesuai dengan aturan dan tidak menabrak pembatas dinding/ <i>station</i> .
7	Wait	Mengetahui apakah dapat menunggu waktu sesuai dengan keinginan tanpa harus menggerakkan simulator.	Menjalankan <i>command</i> WAIT.	Data Test 6.7	Waktu dapat bertambah sesuai dengan input WAIT.	Waktu telah bertambah sesuai dengan input WAIT.
8	Buy, Delivery, Boil, Fry, Chop, dan Mix.	Mengetahui apakah semua <i>command</i> berfungsi dengan baik serta berlaku hanya jika <i>adjacent</i> di <i>station</i> tertentu.	Menjalankan berbagai <i>command</i> tersebut, serta mencoba menjalankan <i>command</i> ketika simulator tidak <i>adjacent</i> dengan <i>station</i> .	Data Test 6.8	<i>Command</i> dapat berjalan dengan baik hanya jika <i>adjacent</i> dengan <i>station</i> yang berkaitan, dan tidak jika jauh dari <i>station</i> .	<i>Command</i> telah berjalan dengan baik hanya jika <i>adjacent</i> dengan <i>station</i> yang berkaitan, dan tidak jika jauh dari <i>station</i> .
9	Inventory	Mengetahui apakah pengguna dapat mengetahui isi <i>inventory</i> .	Menjalankan <i>command</i> INVENTORY.	Data Test 6.9	<i>Inventory</i> dapat ditampilkan sesuai dengan bahan yang dimiliki.	<i>Inventory</i> telah ditampilkan sesuai dengan bahan yang dimiliki.

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
10	Kulkas	Mengetahui apakah kulkas dapat digunakan sesuai dengan fungsionalitasnya yang berlaku.	Menjalankan <i>command</i> kulkas saat adjacent dengan K, menyimpan makanan saat memiliki bahan makanan maupun menyimpan saat <i>inventory</i> kosong, menyimpan bahan makanan saat kulkas telah penuh, mengambil makanan dari kulkas yang terisi maupun mengambil saat kulkas kosong, serta menampilkan isi kulkas.	Data Test 6.10	Fungsi kulkas dapat berjalan sesuai dengan fungsionalitas umumnya.	Fungsi kulkas telah berjalan sesuai dengan fungsionalitas umumnya.
11	Rekomendasi Makanan	Mengetahui apakah program dapat menampilkan resep makanan sesuai dengan bahan yang dimiliki.	Menjalankan <i>command</i> saat memiliki bahan makanan dan saat <i>inventory</i> sedang kosong.	Data Test 11	Rekomendasi dapat muncul sesuai dengan bahan yang dimiliki dan tidak merekomendasikan apapun ketika <i>inventory</i> kosong.	Rekomendasi telah muncul sesuai dengan bahan yang dimiliki dan tidak merekomendasikan apapun ketika <i>inventory</i> kosong.
12	Makanan Kedaluwarsa	Mengetahui apakah notifikasi akan muncul ketika suatu makanan telah mencapai waktu kedaluwarsa.	Menjalankan <i>command</i> WAIT melebihi waktu kedaluwarsa makanan.	Data Test 12	Dapat memunculkan notifikasi terkait makanan yang baru saja kedaluwarsa.	Muncul notifikasi terkait makanan yang baru saja kedaluwarsa.
13	Undo	Mengetahui apakah dapat kembali ke <i>state</i> sebelumnya dengan baik.	Menjalankan <i>command</i> undo saat telah menjalankan sesuatu maupun saat belum	Data Test 13	Dapat kembali ke <i>state</i> sebelumnya jika telah melakukan sesuatu dan gagal undo saat belum.	Dapat kembali ke <i>state</i> sebelumnya jika telah melakukan sesuatu dan gagal undo saat belum.
14	Redo	Mengetahui apakah dapat kembali ke <i>state</i> sebelum di-undo dengan baik.	Menjalankan <i>command</i> redo saat telah menjalankan undo maupun belum meng-undo sesuatu.	Data Test 14	Dapat kembali ke <i>state</i> sebelum di-undo dan gagal redo saat tidak meng-undo apapun.	Kembali ke <i>state</i> sebelum di-undo dan gagal redo saat tidak meng-undo apapun.

8 Pembagian Kerja dalam Kelompok

No	Nama	NIM	Tugas
1	Yanuar Sano Nur Rasyid	13521110	ADT Stack (undo dan redo), melengkapi ADT yang belum, Inisiasi program, testing program.
2	William Nixon	13521123	Menyatukan berbagai ADT agar koheren (command, simulator), Inventory/Delivery, ADT Set (bonus rekomendasi makanan), memperbaiki jika ada kekurangan/bug (undo/redo, parser, kulkas, dll).
3	Nicholas Liem	13521135	ADT Makanan (Catalogue, etc), ADT Kulkas, ADT Time, Parsing File (Makanan, Peta).
4	Akhmad Setiawan	13521164	Notulensi Rapat, Laporan.
5	Reza Pahlevi Ubaidillah	13521165	ADT Peta, Pengolahan makanan (boil, fry, chop, dan mix) beserta waktunya

9 Lampiran

9.1 Deskripsi Tugas Besar 2

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 membuat program simulasi untuk ditiru BNMO. Oleh karena itu, Doni meminta bantuan kalian untuk membuat program simulasi tersebut.

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.





Spesifikasi lengkap dapat dilihat pada link berikut:


[Spesifikasi Tugas Besar IF2110 2022/2023 - Google Docs](#)

9.2 Notulen Rapat

STEI- ITB	IF2110-TB-G-2	Halaman 31 dari 36 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

Asistensi I

Tanggal : 26 Oktober 2022	Catatan Asistensi: <ol style="list-style-type: none">1. Driver tujuannya untuk mengetes implementasi ADT apakah sudah bisa jalan atau belum. Harus menggunakan fungsi yang ada.2. ADT wajib jika ada yang tidak digunakan, dibuang saja, pengecekan bukan via Olympia, nanti dicek manual oleh asisten.3. Strukturnya yang penting menurut kalian memudahkan dan rapi. Fungsi main diletakkan di main.c.4. Penggunaan list statik dan dinamis disesuaikan kegunaannya dari sisi size yang diperlukan. List statik menyimpan daftar resepnya. Jika ingin tahu isi tiap resepnya gunakan list dinamis.5. ADT simulator digunakan untuk menyimpan state.6. Union tujuannya agar diimplementasikan sekali saja tapi bisa digunakan untuk beberapa tipe data (int, char, float). Tidak bisa tiga-tiganya, tapi bisa salah satu dari ketiganya. Referensi: https://www.programiz.com/c-programming/c-unions. Jika tidak gunakan union implementasikan lebih dari sekali. Saran: buat 2 yang berbeda saja.7. Gambar hanya contoh, yang penting program berjalan lancar, terserah kalian mau membuat pengolahan yang seperti apa konfigurasinya, dari bawah pun boleh.8. Boleh buat penanda tersendiri untuk pohon misalkan, parsing terserah kalian konfigurasinya.9. Untuk undo redo lebih enak nge-push / pop command yang udah jalan aja, biar stack-nya ga ngabisin banyak memori.10. Selama di demo tidak ada masalah, tidak terdeteksi ada leak, harusnya tidak masalah, untuk pengaruh ke penilaian tidak perlu tahu. Yang
Tempat : Platform Zoom	
Kehadiran Anggota Kelompok:	
<p>1 13521110</p>  <p>Yanuar Sano Nur Rasyid</p> <p>2 13521123</p>  <p>William Nixon</p> <p>3 13521135</p>  <p>Nicholas Liem</p> <p>4 13521164</p>  <p>Akhmad Setiawan</p>	

	<p>penting fungsionalitas program jalan dengan baik.</p> <ol style="list-style-type: none"> 11. Jika sedang melakukan sesuatu misalkan FRY, boleh jalan-jalan, agar waktunya juga tetap berjalan. 12. 50% laporan maksudnya adalah misal sudah mengerjakan semua fitur, langsung saja tulis di laporannya, yang penting sudah ada progress. 13. Asistensi 2 mengecek progress selanjutnya. 14. <i>Adjacent</i> tidak diagonal. 15. Kalo folder mending lowercase. 16. Saran sih ngerjain di <i>branch</i> masing-masing, tapi terserah kalian, atau repo-nya di-<i>protect</i>.
	<p>Tanda Tangan Asisten:</p> 

Milestone I



Yanuar Sano Nur Rasyid

2
13521123



William Nixon

3
13521135



Nicholas Liem

4
13521164




Akhmad Setiawan

5
13521165



Reza Pahlevi Ubaidillah

7. Modulnya jadikan 1 modul aja, dan selesaikan sisanya, masih ada 19 hari lagi.
8. Akses elemen ga pake selektor gapapa, ga ada ketentuannya.
9. Komentar ga terlalu dicek, ga wajib untuk setiap barisnya.
10. Time ga ada batasnya, terserah kalian mau ngasih batasan atau tidak.
11. Yang susah paling gabunginnya, sisanya sudah aman mungkin.
12. Dicek aja case tertentu yang menurut kalian bisa bikin error untuk di-handle.
13. Asistensi 3 opsional.

	Tanda Tangan Asisten: 
--	--

9.3 Log Activity Anggota Kelompok

No	Nama	NIM	Tanggal	Uraian Kegiatan
1	Semua	Semua	21 Oktober	Membahas bersama spesifikasi laporan, pembagian tugas phase 1 yang terdiri dari: 1. Peta (Parsing dan ADT) - Obed 2. Makanan (Parsing dan ADT) - Nicholas, Iwan 3. Resep (Parsing dan ADT) - William 4. Melengkapi ADT - Sano
2	Semua	Semua	27 Oktober	Persiapan untuk asistensi dan milestone 1, menggabungkan semuanya agar program bisa di run
3	Semua	Semua	2 November	Pembagian tugas phase 2: 1. Kulkas – Nicholas 2. Stack Undo Redo – Sano 3. Inventory, Delivery, Rekomendasi – William, Iwan 4. Masak-masak - Obed
4	Semua	Semua	14 November	Progress update, mengabari satu sama lain jika ada sisa-sisa bug
5	Semua	Semua	18 November	Meeting terakhir, persiapan pengumpulan dan pembetulan bugs, memastikan berjalan di Linux.

Repository Link : Repo: https://github.com/NicholasLiem/IF2110_TugasBesar_BNMO-Simulator-Masak