

LAPORAN TUGAS BESAR
IF2110/Algoritma dan Struktur Data
Program Simulasi Memasak CLI Edition




Dipersiapkan oleh:

Kelompok E Kelas K3

Jason Rivalino	13521008
Hidayatullah Wildan Ghaly B	13521015
Raditya Naufal Abiyu	13521022
M. Malik I. Baharsyah	13521029
Jauza Lathifah Annassalafi	13521030

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		IF2110-TB-E-3		103
		Revisi	0	20 November 2022

Daftar Isi

1 Ringkasan	5
1.1 Deskripsi Umum	6
1.2 Isi Laporan	7
1.3 Kesimpulan	7
2 Penjelasan Tambahan Spesifikasi Tugas	9
2.1 ADT Rekomendasi	9
2.2 Waktu Pengolahan Makanan	9
2.3 Kulkas	10
3 Struktur Data (ADT)	11
3.1 ADT Sederhana	11
3.2 ADT List Statik	11
3.3 ADT Matriks	12
3.4 ADT Mesin Karakter dan Mesin Kata	12
3.5 ADT Queue Food	12
3.6 ADT Stack	13
3.7 ADT Tree	13
3.8 ADT UNDO-REDO	13
3.9 ADT Notifikasi	14
3.10 ADT Stack Integer	14
3.11 ADT Kalimat	14
3.13 ADT Simulator	15
4 Program Utama	16
5 Algoritma-Algoritma Menarik	19
5.1 Command Teleport	19
5.2 Algoritma Toko Tutup	19
5.3 Algoritma Kulkas	19
5.4 Algoritma Hash pada Rekomendasi Makanan	19
6 Data Test	21
6.1 Splash Screen	21
6.1.1 Splash Screen Start	21
6.1.1 Splash Screen Exit	22
6.2 Dataset Peta	23

6.3 Dataset Menu Makanan	23
6.4 Dataset Resep Makanan	25
6.5 Command Inisiasi (START dan EXIT)	29
6.5.1 Start	29
6.5.2 Exit	29
6.6 Command Pemesanan (BUY dan DELIVERY)	30
6.6.1 Buy	30
6.6.2 Delivery	33
6.7 Command Peta (MOVE NORTH, MOVE EAST, MOVE WEST, MOVE SOUTH)	33
6.7.1 Move North	33
6.7.2 Move East	35
6.7.3 Move West	36
6.7.4 Move South	37
6.8 Command Pengolahan (MIX, CHOP, FRY, BOIL)	38
6.8.1 Mix	38
6.8.2 Chop	42
6.8.3 Fry	45
6.8.4 Boil	48
6.9 Command WAIT	51
6.10 Command-Command Lain	52
6.10.1 Undo	52
6.10.1.1 Undo Buy dan Delivery	52
6.10.1.2 Mix dan Inventory	53
Proses MIX berhasil di-undo sehingga inventory terdapat bahan makanan sebelumnya dan tidak terdapat makanan hasil MIX	53
6.10.1.3 Boil dan Inventory	54
6.10.1.4 Fry dan Inventory	55
6.10.1.5 Chop dan Inventory	56
6.10.1.6 Peta	57
6.10.2 Redo	57
6.10.2.1 Buy dan Delivery	57
6.10.2.2 Mix dan Inventory	58
6.10.2.3 Boil dan Inventory	59
6.10.2.4 Fry dan Inventory	60
6.10.2.5 Chop dan Inventory	61
6.10.2.6 Peta	62

6.10.3 Inventory	63
6.10.4 Catalog	64
6.10.5 COOKBOOK	66
6.10.6 Rekomendasi	66
6.10.7 Kulkas	68
7 Test Script	74
7.1 Test Command Inisiasi (START dan EXIT)	74
7.2 Test Command Pemesanan (BUY dan DELIVERY)	74
7.3 Test Command COOKBOOK	76
7.4 Test Command WAIT	77
7.5 Test Command CATALOG	77
7.6 Test Command Pengolahan (MIX, CHOP, FRY, BOIL)	78
7.7 Test Command Peta (MOVE NORTH, MOVE EAST, MOVE WEST, MOVE SOUTH)	91
7.8 Test Command Rekomendasi (BONUS)	92
7.9 Test Command Kulkas (BONUS)	93
8 Pembagian Kerja dalam Kelompok	95
9 Lampiran	95
9.1 Deskripsi Tugas Besar	95
9.2 Notulen Rapat	96
9.2.1 Asistensi 1	96
9.2.2 Notulen Rapat 1	98
9.2.3 Notulen Rapat 2	99
9.2.4 Asistensi 2	101
9.2.5 Notulen Rapat 3	102
9.3 Log Activity Anggota Kelompok	103
9.4 Milestone 1	109

1 Ringkasan

1.1 Deskripsi Umum

Pada tugas besar kali ini, kami diminta untuk membuat sebuah program simulasi memasak berbasis **CLI** (*command-line interface*) yang dibuat dalam bahasa C. Program dimulai dengan pengguna menginisiasi aplikasi yang akan menampilkan splash screen sebelum program meminta *command* pertama (START/EXIT) kepada pengguna. Ketika *command* START dijalankan, program akan memiliki seluruh informasi yang dibutuhkan, seperti *layout* peta dan isinya, informasi makanan yang valid, dan resep yang valid. Pada program ini, pengguna dapat melakukan pemesanan makanan, mengelola makanan menggunakan beberapa command (mix, chop, fry, dan boil), melihat bahan dan makanan yang tersedia, menampilkan resep, melihat makanan yang ada dimiliki di inventory, melihat makanan yang sedang diantar, menampilkan ketika bahan makanan yang dipesan sudah sampai, membuang makanan yang sudah mencapai waktu kedaluarsa. Pengguna juga dapat menggunakan fitur-fitur lain yang tersedia sesuai spesifikasi tugas besar.

Ada beberapa ADT yang digunakan, yaitu:

1. ADT Sederhana
2. ADT List Statik
3. ADT Matriks
4. ADT Mesin Karakter dan Mesin Kata
5. ADT Queue dengan Pendekatan Array List Dinamik
6. ADT Stack
7. ADT Tree
8. ADT Undo-Redo
9. ADT Notifikasi
10. ADT Stack Integer
11. ADT Kalimat
12. ADT Rekomendasi
13. ADT Simulator

Penjelasan mengenai ADT-ADT di atas terdapat pada BAB III Struktur Data.

STEI- ITB	IF2110-TB-E-3	Halaman 6 dari 110 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.		

1.2 Isi Laporan

Laporan ini berisi penjelasan mengenai program yang telah kami buat. Bagian pertama laporan dijelaskan ringkasan dari laporan ini yang berisikan deskripsi umum, isi laporan secara singkat, dan kesimpulan dari hasil tugas besar yang telah dibuat. Bagian kedua terdiri dari penjelasan tambahan mengenai spesifikasi-spesifikasi fitur yang belum rinci dari deskripsi tugas besar. Bagian ketiga berisi berbagai struktur data yang dibuat dan digunakan. Bagian keempat berisi tentang penjelasan mengenai program utama. Bagian kelima berisi algoritma-algoritma menarik yang digunakan dalam program. Bagian keenam terdiri dari data test dan test script dari program yang telah dibuat. Bagian kedelapan berisi tentang pembagian kerja dalam kelompok dan lampiran-lampiran. Bagian terakhir berisi lampiran yang terdiri dari deskripsi tugas besar, notulen-notulen rapat, log activity kelompok, dan *progress* dari pengerjaan tugas besar kepada asisten pembimbing.

1.3 Kesimpulan

Pada tugas besar kali ini kami berhasil membuat sebuah program simulasi memasak dalam CLI menggunakan bahasa C. Dalam program ini, kami menggunakan beberapa *Abstract Data Type* (ADT) Seperti ADT list statik dan dinamik, ADT *queue* dan *stack*, ADT *tree*, ADT *word machine* dan *char machine*, ADT *tree*, dan beberapa ADT tambahan. Program dimulai dengan meminta masukan pengguna mengenai nama pengguna kemudian dilanjutkan dengan inisiasi dan pada akhirnya masuk ke tahap simulasi memasak yang bisa dimainkan pengguna dengan memasukkan perintah ke dalam CLI yang telah disediakan. Setelah puas bermain pengguna dapat menggunakan perintah EXIT untuk keluar dari permainan simulasi memasak ini.

Menurut kelompok kami, tugas besar yang diberikan sangat membantu kami untuk memahami kegunaan berbagai ADT dan bagaimana setiap ADT bekerja bersama untuk membangun suatu program. Tugas besar ini juga membantu kami untuk belajar melakukan manajemen mulai dari manajemen waktu hingga pembagian tugas sesuai kemampuan masing-masing anggota kelompok. Namun pada program kami, terkadang muncul *bug* pada beberapa perangkat dan sistem operasi namun *bug* tersebut tidak mengganggu keberjalanan

permainan simulasi memasak ini. Terakhir dari kami, mohon maaf apabila ada ketidak sempurnaan dalam program yang kami buat karena kesempurnaan hanya milik Tuhan.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 ADT Rekomendasi

ADT Rekomendasi merupakan ADT dengan memanfaatkan struktur data stack dan list hash. ADT ini hanya berisikan integer 0 dan 1 dengan panjang list 200 kecuali elemen pertamanya. Elemen pertama atau index ke 0 dari ADT Rekomendasi ini merupakan ID dari makanan yang akan dibuat. Apabila ADT ini digunakan pada inventory maka elemen pertamanya bernilai nol. Elemen selanjutnya dari ADT ini berisi 0 dan 1 dengan 0 merupakan ID yang tidak dimiliki oleh list dan 1 merupakan ID yang dimiliki list. ADT Rekomendasi dimanfaatkan untuk mendapatkan makanan rekomendasi berdasarkan inventory. Apabila sebuah makanan terbuat dari beberapa makanan lainnya maka ADT Rekomendasinya berbentuk elemen pertama bernilai ID makanan tersebut dan elemen berikutnya merupakan bahan yang diperlukan untuk membuat makanan tersebut. Hal yang mirip terjadi untuk inventory, ADT Rekomendasi akan menyimpan semua ID yang ada pada inventory sebagai 1 pada indeks ID inventory.

Cara program mendapatkan rekomendasi makanan yang bisa dibuat oleh user adalah dengan membandingkan ADT Rekomendasi pada inventory dan ADT Rekomendasi pada setiap resep yang ada. Apabila ADT Rekomendasi pada suatu resep tertentu merupakan subset dari ADT Rekomendasi pada inventory maka resep tersebut akan direkomendasikan untuk user. Dengan begitu akan didapatkan semua rekomendasi yang bisa dibuat berdasarkan semua resep yang ada.

2.2 Waktu Pengolahan Makanan

Waktu pengolahan makanan merupakan waktu yang dilewati saat makanan diproses. Pemrosesan makanan sendiri memiliki empat tipe yaitu MIX, CHOP, FRY, dan juga BOIL. Setiap proses memiliki pemrosesan yang berbeda. MIX memiliki waktu pemrosesan tiga menit, CHOP memiliki waktu pemrosesan empat menit, FRY memiliki waktu pemrosesan lima menit, dan BOIL memiliki waktu pemrosesan 6 menit.

Apabila simulator mulai untuk melakukan salah satu dari empat pemrosesan makanan, maka simulator akan berada di tempat tersebut selama waktu yang telah ditentukan saat pemrosesan makanan. Hal ini dikarenakan simulator harus berfokus dalam pemrosesan makanan

tersebut sehingga tidak memungkinkan baginya untuk berpindah ke tempat lain di saat melakukan pemrosesan makanan. Untuk pengguna, tampilan yang terlihat adalah waktu simulator yang berubah dan juga hasil yang dicapai dari pemrosesan makanan.

2.3 Kulkas

Kulkas merupakan sebuah penyimpanan dalam bentuk matriks dengan ukuran 20 x 20 yang menyimpan elemen tipe makanan dan boolean. Elemen makanan pada kulkas digunakan untuk menyimpan makanan tersebut dalam kulkas. Makanan yang disimpan dalam kulkas tersebut akan memberikan info boolean pada indeks-indeks matriks yang telah menyimpan suatu makanan. Apabila sudah ada makanan pada indeks matriks tertentu maka info boolean yang ada pada indeks tersebut bernilai true, sedangkan jika tidak ada makanan pada indeks tersebut maka info booleannya bernilai false.

3 Struktur Data (ADT)

ADT yang dipergunakan oleh kelompok kami dalam membangun program antara lain:

3.1 ADT Sederhana

ADT Sederhana merupakan ADT yang digunakan untuk membentuk struktur data Point, Waktu, Makanan, dan Simulator. Pada program, ADT Point memiliki dua buah variabel bertipe data integer, yaitu X dan Y yang digunakan untuk mempresentasikan lokasi suatu benda pada bidang dua dimensi. ADT Waktu memiliki data Hari, Jam, dan Menit. ADT ini digunakan untuk menambahkan durasi waktu setiap *command* perpindahan yang dilakukan oleh BNMO, setiap *command* yang terkait pengolahan makanan (BUY, MIX, CHOP, FRY, BOIL), dan keberjalanan waktu ini juga mengurangi waktu *delivery* dari makanan yang dipesan. ADT Makanan terdiri dari id makanan, nama makanan, waktu kedaluwarsa, lokasi aksi makanan, dan lama pengiriman makanan. ADT Simulator digunakan untuk representasi BNMO pada program yang memiliki beberapa informasi yang disimpan, yaitu nama pengguna simulator, lokasi simulator di peta, dan inventory makanan yang disimpan.

3.2 ADT List Statik

ADT List Statik merupakan ADT yang digunakan untuk menyimpan data-data makanan yang telah ada. Pada program, ADT ini kemudian akan dipakai untuk menampilkan makanan pada command CATALOG dan juga untuk menyalin makanan ke inventory. Procedure dan fungsi yang ada pada ADT ini yaitu procedure primitif ADT List Statik untuk membuat list statik, lalu ada fungsi-fungsi untuk menghitung panjang list statik, mencari indeks pertama dan terakhir untuk list statik, mengecek indeks yang valid dan terdefinisi untuk list statik, mengecek apakah list statik kosong dan penuh, mengecek kesamaan list statik, mengecek elemen list dengan nilai tertentu, serta untuk memasukkan dan menghapus elemen pada list statik. Untuk aplikasi dari fungsi primitifnya, ada membuat fungsi baru untuk mencari makanan dari IDnya dalam bentuk type word dan makanan serta mengembalikan nilai boolean berdasarkan pengecekan, ada fungsi untuk membaca file makanan dan memberikan keluaran makanan tipe BUY. Selain fungsi, ada procedure juga untuk sorting list statik untuk makanan, mencetak list statik, dan juga procedure CATALOG untuk menampilkan daftar makanan yang akan digunakan untuk commandnya.

3.3 ADT Matriks

ADT Matriks merupakan ADT yang digunakan untuk menyusun dan membuat sebuah matriks. Pada program, ADT ini atau ADT MAP merupakan ADT yang digunakan sebagai struktur untuk membuat peta yang akan digunakan sebagai navigasi BNMO untuk menjalankan aktivitas tertentu. Pada program, tujuan output akhir pada ADT ini adalah untuk menghasilkan gambar peta dalam bentuk matriks. ADT ini akan menerima dan membaca data dalam bentuk nama file. Kemudian ADT ini akan memanfaatkan implementasi ADT lainnya, yaitu ADT Mesin Kata untuk membaca string yang ada pada file. Pembacaan dimulai dari membaca ukuran matriks $M \times N$ yang kemudian hasil ukurannya ini akan diubah dari bentuk string ke dalam bentuk integer. Lalu program akan memanfaatkan ADT Matriks untuk membaca string dan memasukkannya ke dalam bentuk matriks hingga mencapai *endword*. Bentuk akhir yang dikembalikan adalah matriks yang sudah jadi dan disimpan dalam variabel matriks.

3.4 ADT Mesin Karakter dan Mesin Kata

ADT Mesin Karakter dan Mesin Kata merupakan ADT yang digunakan sebagai pembaca masukkan berupa teks yang terdiri dari dua karakter atau lebih. Pada program, kedua buah ADT ini digunakan untuk membantu membaca file .txt yang sudah dibuat lalu hasil bacaannya akan digunakan untuk program. Procedure yang ada pada ADT Mesin Karakter antara lain adalah Procedure untuk memulai membaca pita, memulai operasi mesin karakter, dan memajukan pita karakter. Sedangkan, pada ADT Mesin Kata, procedure yang ada antara lain untuk cek spasi, memulai kata, membuat kata kosong, menyalin kata, menyimpan kata, dan menampilkan kata.

3.5 ADT Queue Food

ADT Queue Food merupakan ADT yang memanfaatkan struktur data Priority Queue Time. ADT ini digunakan untuk menyimpan makanan dengan pendekatan priority queue time. Hal ini digunakan agar makanan bisa dibatasi selama apa mereka bisa berada pada struktur data queue time ini. ADT ini dimanfaatkan untuk proses delivery dan juga inventory. Priority time yang digunakan adalah nilai waktu saat ini (real time) yang ditambah dengan apa yang diperlukan. Untuk delivery maka priority time yang digunakan adalah waktu saat ini ditambah dengan delivery time dan untuk inventory maka priority time yang digunakan adalah waktu saat ini yang ditambah dengan expired time. Hal ini dilakukan dengan tujuan agar pengecekan bisa langsung dilakukan dengan cara jika real time sudah melebihi waktu pada peiority queue maka makanan tersebut akan dikeluarkan dari list.

3.6 ADT Stack

ADT Stack merupakan ADT yang digunakan untuk meng-undo atau meng-redo *command* yang dilakukan oleh simulator pada aplikasi. ADT Stack digunakan untuk menyimpan suatu state/keadaan sebelum sebuah perintah dimasukkan ke dalam program oleh pengguna. Ketika pengguna memasukkan perintah Undo, program akan mengambil state yang tersimpan paling atas pada Stack lalu merubah semua kondisi menjadi kondisi state yang telah diambil dari Stack. Berlaku juga apabila makanan selesai diantar atau kadaluarsa.

3.7 ADT Tree

ADT Tree merupakan ADT yang digunakan untuk menyimpan resep dalam bentuk graf pohon agar mempermudah pengguna dalam mendapatkan resep dari suatu makanan. Tree sendiri memiliki satu parent yang terdiri atas satu atau lebih child yang dalam hal ini child bisa saja tidak memiliki child atau memiliki satu atau lebih child. Child yang sudah tidak memiliki anakan ini dapat diartikan bahwa child tersebut tidak bisa didapatkan dari mencampurkan bahan tertentu, melainkan harus membelinya dengan command BUY.

3.8 ADT UNDO-REDO

ADT Undo-Redo merupakan ADT yang menyimpan struktur real time untuk menyimpan proses di waktu tertentu, simulator untuk menyimpan isi dari inventory di saat tersebut, dan juga queue delivery untuk menyimpan isi dari makanan yang sedang dikirim. Proses dari undo redo sendiri adalah dengan memanfaatkan stack yang berisikan ADT undo-redo ini. Proses yang bisa di-Undo akan menjadikan variabel isValid menjadi true dan proses saat itu akan di push pada stack utama dari Undo-Redo. Apabila proses ini di undo, maka stack akan di pop lalu hasilnya disimpan di stack redo. Proses yang mirip terjadi pada command redo yang melakukan pop apabila dijalankan command redo dan hasil popnya akan disimpan di stack utama undo-redo. Apabila proses yang dilakukan valid, proses disimpan di stack utama undo-redo, di saat yang sama, stack redo akan dikosongkan karena sudah tidak dibutuhkan lagi. Pengambilan proses dari stack utama undo-redo adalah dengan melakukan copy pada kepala dari stack utama undo-redo.

Stack utama undo-redo tidak mungkin kosong. Pada saat pertama kali dijalankan, waktu akan di set pukul 00.00.00 yang berarti hari ke-0, jam ke-0, dan menit ke-0. Di saat yang sama, nama simulator akan langsung diminta oleh program agar program dapat berjalan. Dengan

adanya nama simulator, maka simulator akan terbentuk dengan memiliki inventory kosong dan peta simulator akan mulai dibaca. Dengan begitu, simulator akan siap untuk disimpan ke dalam stack utama undo-redo. Selain itu, di awal juga akan dilakukan set queue delivery kosong yang juga akan disimpan dalam stack utama undo-redo. Dengan persiapan ini, maka proses utama pertama telah terbentuk dan siap untuk di push ke stack utama undo-redo. Dengan begitu, stack utama undo redo akan berisi setidaknya 1 buah proses dan proses ini tidak bisa di undo lagi.

3.9 ADT Notifikasi

ADT Notifikasi merupakan ADT yang memanfaatkan struktur data stack untuk menyimpan notifikasi hasil dari proses yang telah dilakukan. Notifikasi ini menyimpan notifikasi dalam bentuk struktur data word. Apabila proses telah selesai dilakukan, maka akan ada push notifikasi pada stack notifikasi untuk diproses kemudian. Stack notifikasi akan selalu dikosongkan di awal program berjalan. Di saat stack notifikasi tidak kosong, maka akan dilakukan display notifikasi sekaligus pengosongan notifikasi itu dengan menggunakan loop. Display pada notifikasi tidak bisa di undo, tetapi display akan tetap ditampilkan pada saat dijalankan command redo karena proses redo merupakan proses maju yang memungkinkan adanya notifikasi baru.

3.10 ADT Stack Integer

ADT Stack Integer merupakan ADT dengan memanfaatkan struktur data stack dan memiliki elemen-elemen integer. ADT Stack integer ini digunakan untuk membantu perhitungan pada charmachine, wordmachine, dan sentencemachine apabila inputnya merupakan sebuah string. Pada main, input selalu dalam bentuk kalimat, dengan kata lain harus ada pemisah antara input kalimat tersebut dengan input integer. Salah satu pemanfaatan stack integer ini adalah pada command WAIT X Y yang menerima 'wait' sebagai command serta X dan Y merupakan suatu integer. Dengan begitu, stack integer akan membantu pengerjaan agar bisa memisah kalimat dengan integer X dan Y agar bisa diproses dalam command wait.

3.11 ADT Kalimat

ADT Kalimat merupakan ADT dengan memanfaatkan struktur mesin kata. ADT ini memiliki struktur array of char yang merupakan tab kalimat dan juga panjang dari kalimat tersebut. ADT Kalimat dimanfaatkan untuk main dari program agar pengguna dapat menginput

string dengan panjang yang bebas. ADT Kalimat akan melakukan validasi terhadap input pengguna dan akan menampilkan input tidak valid apabila input pengguna tidak terdaftar dalam daftar command. Apabila pengguna melakukan input yang terdaftar pada command maka program akan melakukan validasi dengan fungsi `isKalimatEqual`. Apabila `isKalimatEqual` me-return true maka program akan memanggil fungsi command yang telah tervalidasi.

3.13 ADT Simulator

ADT Simulator merupakan ADT yang menyimpan struktur data Word sebagai nama pengguna, Matrix sebagai peta untuk mengetahui lokasi pengguna, dan queue food untuk menyimpan inventory pengguna. Simulator akan memanfaatkan waktu utama (realTime) untuk pengecekan inventorynya. Inventory pada simulator berbentuk queue food yang dalam hal ini akan memiliki waktu expired yang di set tepat setelah makanan masuk ke dalam inventory. Waktu expired yang disimpan dalam inventory merupakan waktu saat makanan masuk ke dalam inventory yang ditambah dengan waktu makanan expired tersebut.

Apabila simulator membeli makanan A yang memiliki expired time 15 menit dan waktu delivery 3 menit, maka makanan akan berada di delivery selama 3 menit. Jika waktu saat simulator membeli makanan A adalah pukul 10.00 maka makanan akan sampai pada pukul 10.03 dan akan masuk inventory. Tepat setelah makanan masuk ke dalam inventory, expired time dari makanan akan di set pukul 10.18. Jadi, makanan akan expired apabila real time sudah mencapai 10.18 atau lebih.

4 Program Utama

Program Utama atau main merupakan program yang digunakan untuk menjalankan program yang sudah mencakup semua fungsi yang diperlukan. Program ini terletak di folder Apps dan bernama MAIN.c. Pada bagian paling teratas main dilakukan include terhadap semua file yang diperlukan. Setelah melakukan include ke seluruh file yang diperlukan, maka program utama siap dibuat di dalam fungsi `int main()`.

Pada bagian pertama main, dipanggil fungsi `splash()` yang dipakai untuk menampilkan splash screen awal. Setelah penampilan splash screen maka akan dilanjutkan dengan kalimat “Welcome to Diner Dash CLI Edition!”. Setelah penampilan tersebut pada layar, program akan mulai membaca file yang diperlukan. File pertama yang dibaca adalah daftar makanan yang akan disimpan ke dalam variabel **daftarMakanan** lalu dilanjutkan dengan daftar makanan yang hanya bisa didapatkan dengan cara buy lalu disimpan dalam variabel **buyAbleFood**.

Setelah program mendapatkan daftar makanan, program akan mulai membaca daftar resep yang akan disimpan ke dalam variabel **daftarResep** yang merupakan list yang berisi struktur data pohon. Selain menyimpannya ke dalam `daftarResep`, program juga akan membuat daftar resep dalam bentuk advanced yang berarti resep yang merupakan bahan dari resep lainnya tidak bisa diakses kecuali dengan mengakses resep utamanya dan bentuk advanced resep ini disimpan dalam variabel **advancedResep**.

Selain pembacaan makanan, program juga akan membaca bentuk peta dari file lalu membentuknya menjadi peta yang siap untuk dioperasikan. Peta ini akan disimpan dalam variabel **peta**. Setelah semua bahan yang diperlukan sudah siap untuk disimulasikan, maka sudah saatnya program membentuk simulator **sim** dengan menerima input nama simulator. Simulator ini akan dibentuk setelah pengguna menginput nama dari simulator tersebut. Simulator ini terbentuk dari nama yang dimasukkan sebagai namanya, peta yang dimasukkan sebagai lokasinya, dan juga priority queue kosong sebagai inventorynya.

Adanya simulator yang telah dibentuk, tentunya memerlukan sebuah penanda waktu untuk menandai waktu saat itu. Dengan begitu, program mulai membuat waktu dengan set hari ke-0, jam ke-0, dan menit ke-0 sebagai awal dari simulator. Waktu ini diyakini sebagai waktu utama dan disimpan ke dalam variabel **realTime**.

Saat ini bentukan simulator telah selesai dan siap untuk dioperasikan. Untuk mempermudah pengoperasian dalam program utama ini, program menyimpan beberapa variabel pembanding untuk mengecek validasi dari input pengguna. Variabel pembanding tersebut diantaranya adalah start, exit, buy, delivery, mix, chop, boil, fry, undo, redo, catalog, cookbook, rekomendasi, kulkas, dan inventory. Dengan adanya variabel pembanding, maka sekarang program sudah siap untuk menerima input pengguna dalam bentuk string dan input pengguna sudah bisa divalidasi walaupun inputan salah atau random.

Dengan persiapan yang sudah ada, sekarang program hanya perlu membuat penyempurnaan yang diperlukan untuk membantu keberjalanan program. Program akan mulai mendeklarasi stack rekomendasi makanan yang akan disimpan dalam variabel **Rekomendasi_Makanan**. Setelah itu program juga akan mendeklarasi dan membentuk priority queue time untuk delivery agar bisa menyimpan makanan yang sedang diantar di dalam variabel **DELIVERY**.

Simulator ini merupakan simulasi, sehingga diperlukan proses undo dan redo apabila pengguna melakukan sebuah kesalahan. Untuk itu, program mulai melakukan deklarasi sekaligus pembuatan stack undo dan juga stack redo. Stack undo akan digunakan untuk menyimpan proses utama yang terjadi pada simulator sedangkan stack redo akan digunakan untuk menyimpan proses yang telah di-undo oleh pengguna. Stack undo disimpan dalam variabel **stack_UTAMA** sedangkan stack redo disimpan dalam variabel **stack_REDO**. Selain pembuatan stack tersebut, saat ini sudah ada proses yang terjadi yaitu proses tahap nol. Proses ini menyimpan simulator, waktu utama, dan juga delivery. Proses saat ini memiliki waktu 00:00:00, dengan begitu itu merupakan proses pertama yang akan berada di stack_UTAMA. Proses ini disimpan dalam variabel **proses** yang berbentuk elemen dari stack undo redo.

Simulator masih kurang lengkap tanpa adanya notifikasi. Untuk itu, program membentuk stack notifikasi yang akan digunakan untuk menampilkan notifikasi yang terjadi. Stack notifikasi sendiri memiliki elemen berupa kata (word) dan notifikasi akan selalu dikosongkan setelah ditampilkan ke layar. Stack notifikasi ini disimpan ke dalam variabel **Notifikasi**. Untuk membantu pemrosesan pada command **WAIT**, program mendeklarasi dua buah integer yang disimpan dalam **hh** dan **mm** untuk menyimpan angka yang diinput oleh pengguna setelah memanggil fungsi WAIT. Selain membantu pengguna, program juga melakukan deklarasi

boolean terhadap variabel **frontNotif** dan **backNotif**. Kedua variabel ini digunakan untuk memberi tahu program apakah saat itu program sedang dalam keadaan maju atau mundur (di-undo).

Sekarang program sudah benar-benar siap untuk dijalankan. Sekarang program akan meminta pengguna untuk melakukan input **START** untuk memulai program. Apabila pengguna menginput **EXIT** maka program akan langsung keluar tanpa menjalankan program utama, sedangkan jika pengguna menginput selain **START** dan **EXIT** maka program akan meminta input ulang dari pengguna sampai inputnya valid. Jika pengguna telah memutuskan untuk melakukan **START** maka sekarang program utama akan benar-benar mulai berjalan.

Setelah program utama telah benar-benar berjalan maka akan terjadi loop tanpa henti sampai pengguna melakukan input **EXIT**. Proses dalam loop yang pertama dilakukan program adalah menampilkan nama simulator, posisi simulator, waktu saat ini, peta simulator, dan juga notifikasi. Setelah tampilan tersebut pada layar maka program akan meminta input pengguna. Apabila input pengguna tidak terdaftar, maka program akan menampilkan input tidak valid. Apabila input terdaftar, maka program akan menjalankan fungsi yang dipanggil oleh pengguna. Fungsi yang dijalankan oleh program memiliki penjelasan lebih rinci pada bagian 6.

5 Algoritma-Algoritma Menarik

Algoritma-algoritma menarik yang dipergunakan oleh kelompok kami dalam membangun program antara lain:

5.1 Command Teleport

Dalam program yang kelompok kami buat, kami menambahkan command Teleport yang memiliki fungsi yaitu untuk langsung memindahkan simulator BNMO langsung ke koordinat yang ingin dituju. Command teleport ini bisa dipanggil dengan cara menuliskan “TELEPORT X Y” dengan X menyatakan posisi nilai baris dan Y menyatakan posisi nilai kolom yang ingin dituju oleh simulator BNMO. Jika posisi berada diluar nilai matriks yang valid ataupun titik yang ingin dituju pada matriks sudah terdapat fitur menu, program akan menampilkan pesan gagal berpindah. Jika lokasi yang dimasukkan valid, simulator BNMO akan langsung berpindah menuju lokasi yang diinputkan dengan tambahan waktu dalam program sebanyak 10 menit.

5.2 Algoritma Toko Tutup

Dalam program yang kami buat, kami menambahkan fitur realistik yaitu dari jam 22.00 hingga 08.00 toko akan tutup dan tidak bisa diakses. Hal ini mengikuti jadwal buka kebanyakan toko di sekitar kita sehingga pemain dapat merasakan realisme dan lebih peduli terhadap waktu saat bermain permainan yang kami buat.

5.3 Algoritma Kulkas

Kulkas merupakan sebuah penyimpanan dalam bentuk matriks dengan ukuran 20 x 20 yang menyimpan elemen tipe makanan dan boolean. Elemen makanan pada kulkas digunakan untuk menyimpan makanan tersebut dalam kulkas. Makanan yang disimpan dalam kulkas tersebut akan memberikan info boolean pada indeks-indeks matriks yang telah menyimpan suatu makanan. Apabila sudah ada makanan pada indeks matriks tertentu maka info boolean yang ada pada indeks tersebut bernilai true, sedangkan jika tidak ada makanan pada indeks tersebut maka info booleannya bernilai false.

5.4 Algoritma Hash pada Rekomendasi Makanan

Rekomendasi makanan pada program memanfaatkan algoritma hash dengan membandingkan dua buah stack. Dua buah stack yang dibandingkan merupakan stack yang

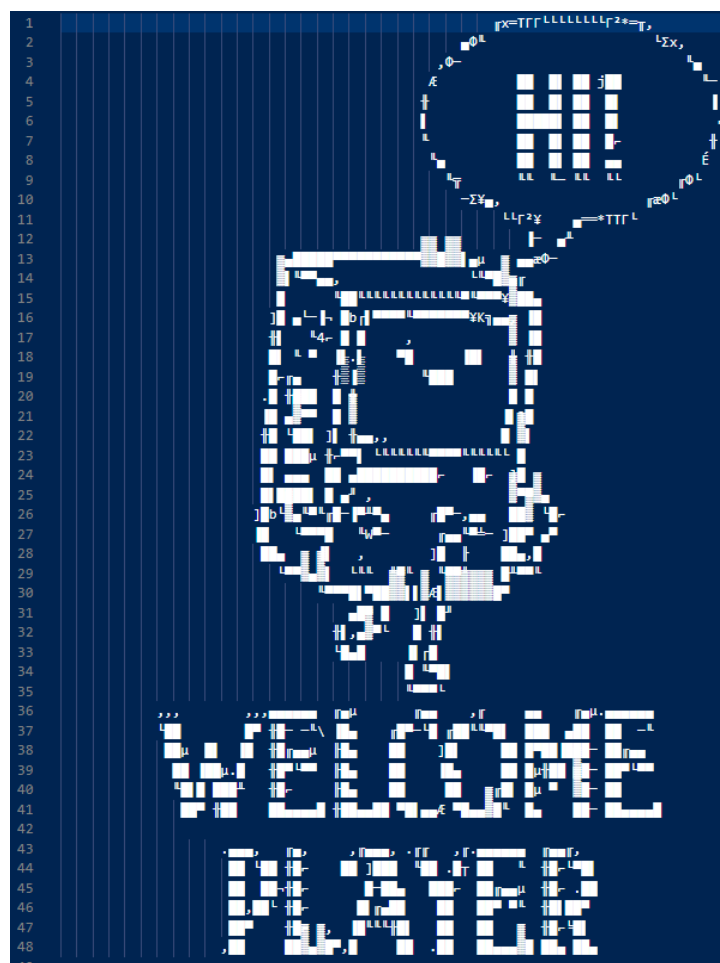
berisi elemen-elemen integer 0 dan 1 kecuali pada indeks pertamanya. Konsep ini dimanfaatkan untuk menemukan fakta apakah sebuah stack merupakan subset dari stack lainnya sehingga bisa dijadikan rekomendasi jika benar.

6 Data Test

6.1 Splash Screen

6.1.1 Splash Screen Start

Pada fitur splash screen start ini, program akan menampilkan splash screen yaitu berupa character BNMO dengan tulisan selamat datang untuk user dari program ini. Setelah splash screen muncul, program akan meminta nama dari user dan proses program pun akan dimulai.



Gambar 6.1.1.1 Tampilan Splash Screen awal

6.1.1 Splash Screen Exit

Pada fitur splash screen exit ini, setelah user memasukkan input “EXIT”, program akan menampilkan splash screen yaitu berupa character BNMO dengan tulisan ucapan terima kasih untuk user dari program ini dan program pun akan berhenti.



Gambar 6.1.2.1 Tampilan Splash Screen akhir

6.2 Dataset Peta

Pada dataset peta, program akan menerima dataset peta yang telah dibuat terlebih dahulu sebelumnya dalam file .txt dengan baris pertama adalah iterasi untuk jumlah baris dan kolom yang akan dibentuk matriksnya. Lalu isi dari baris berikutnya adalah berupa matriks yang ingin dibentuk dengan tanda pagar menunjukkan posisi yang tidak terisi dari map, lalu ada huruf-huruf yang menandakan posisi command dan kemudian diakhiri dengan titik sebagai penutup dari file peta yang akan dibaca. Output akhir yang akan dihasilkan berupa gambar peta navigasi BNMO.

```
1  15 15
2  S#####
3  #####T##X#X#X#
4  #####XXX#X#
5  #####X#X#X#
6  ###M#####
7  #X#####
8  #X#####C###
9  #XXX##F#####
10 #####
11 ##B#####XXX#
12 #####X####
13 #####XXX##
14 #####K####X####
15 #####XXX#
16 #####.
```

Gambar 6.2.1 Tampilan File .txt untuk matriks peta

6.3 Dataset Menu Makanan

Pada dataset menu makanan, program akan menerima dataset menu makanan yang telah dibuat terlebih dahulu sebelumnya dalam file .txt dengan baris pertama adalah iterasi untuk jumlah total jenis makanan yang ada dalam program. Berikutnya, untuk setiap jenis makanan,

terdapat info berupa ID Makanan, nama dari makanan, lama waktu makanan kadaluarsa, waktu pengiriman, dan juga aksi untuk mendapatkan makanan ini. Dataset yang ada ini kemudian akan dibaca dan digunakan untuk berbagai macam fitur dalam program seperti untuk menentukan waktu pengolahan, waktu pengiriman, dan juga iterasi proses untuk membuat makanan yang diperlukan.

36	Buy	71	Buy	1	47	106	Fry
37	8	72	15	2	1	107	22
38	Bawang	73	Teh Sachet	3	Beras	108	Ayam Potong
39	0 12 0	74	35 0 0	4	30 0 0	109	1 5 0
40	0 0 5	75	0 0 5	5	0 0 20	110	0 0 0
41	Buy	76	Buy	6	Buy	111	Chop
42	9	77	16	7	2	112	23
43	Tahu Mentah	78	Kopi Sachet	8	Santan	113	Ayam Tepung
44	5 10 0	79	35 0 0	9	12 0 0	114	0 15 0
45	0 0 20	80	0 0 5	10	0 0 10	115	0 0 0
46	Buy	81	Buy	11	Buy	116	Mix
47	10	82	17	12	3	117	24
48	Tempe Mentah	83	Es Batu Bongkahan	13	Kunyit	118	Ayam Goreng
49	5 10 0	84	0 0 30	14	17 15 0	119	0 12 0
50	0 0 20	85	0 0 5	15	0 0 5	120	0 0 10
51	Buy	86	Buy	16	Buy	121	Fry
52	11	87	18	17	4	122	25
53	Telur	88	Nasi Putih	18	Tepung	123	Sambal
54	5 10 0	89	1 12 0	19	25 10 0	124	1 12 0
55	0 0 5	90	0 0 35	20	0 0 10	125	0 0 0
56	Buy	91	Boil	21	Buy	126	Mix
57	12	92	19	22	5	127	26
58	Bakso Mentah	93	Nasi Kuning	23	Minyak Goreng	128	Sambal Goreng
59	8 20 0	94	1 0 0	24	10 0 0	129	0 10 0
60	0 0 25	95	0 0 15	25	0 0 15	130	0 0 5
61	Buy	96	Mix	26	Buy	131	Fry
62	13	97	20	27	6	132	27
63	Sosis Mentah	98	Nasi Uduk	28	Ayam Mentah	133	Tahu Goreng
64	8 20 0	99	1 0 0	29	22 0 0	134	0 20 0
65	0 0 25	100	0 1 0	30	0 0 30	135	0 0 8
66	Buy	101	Mix	31	Buy	136	Fry
67	14	102	21	32	7	137	28
68	Air	103	Nasi Goreng	33	Cabai	138	Tempe Goreng
69	50 0 0	104	0 18 0	34	3 0 0	139	0 20 0
70	0 0 5	105	0 0 20	35	0 0 5	140	0 0 8

141	Fry	176	Mix	211	Mix
142	29	177	36	212	43
143	Telur Dadar	178	Kopi Panas	213	Nasi Uduk dengan Ayam Goreng Extra Sambal
144	0 16 0	179	0 0 25	214	0 8 0
145	0 0 7	180	0 0 2	215	0 0 20
146	Fry	181	Mix	216	Mix
147	30	182	37	217	44
148	Telur Rebus	183	Es Batu	218	Nasi Goreng dengan Bakso
149	1 21 30	184	0 0 28	219	0 15 30
150	0 0 25	185	0 0 0	220	0 0 20
151	Boil	186	Chop	221	Mix
152	31	187	38	222	45
153	Telur Balado	188	Es Teh	223	Nasi Goreng dengan Sosis
154	0 15 30	189	0 0 25	224	0 15 30
155	0 0 5	190	0 0 2	225	0 0 20
156	Mix	191	Mix	226	Mix
157	32	192	39	227	46
158	Bakso Goreng	193	Kopi Es	228	Nasi Goreng dengan Telur Dadar
159	2 3 0	194	0 0 25	229	0 15 30
160	0 0 0	195	0 0 2	230	0 0 20
161	Fry	196	Mix	231	Mix
162	33	197	40	232	47
163	Sosis Goreng	198	Ayam Goreng Extra Sambal	233	Nasi Kuning dengan Telur Balado
164	2 3 0	199	0 11 30	234	0 8 0
165	0 0 0	200	0 0 15	235	0 0 20
166	Fry	201	Mix	236	Mix.
167	34	202	41		
168	Air Panas	203	Nasi Putih dengan Ayam Goreng Extra Sambal		
169	0 0 30	204	0 11 0		
170	0 0 0	205	0 0 20		
171	Boil	206	Mix		
172	35	207	42		
173	Teh Panas	208	Nasi Kuning dengan Ayam Goreng Extra Sambal		
174	0 0 25	209	0 8 0		
175	0 0 2	210	0 0 20		

Gambar 6.3.1 Tampilan File .txt untuk dataset menu makanan

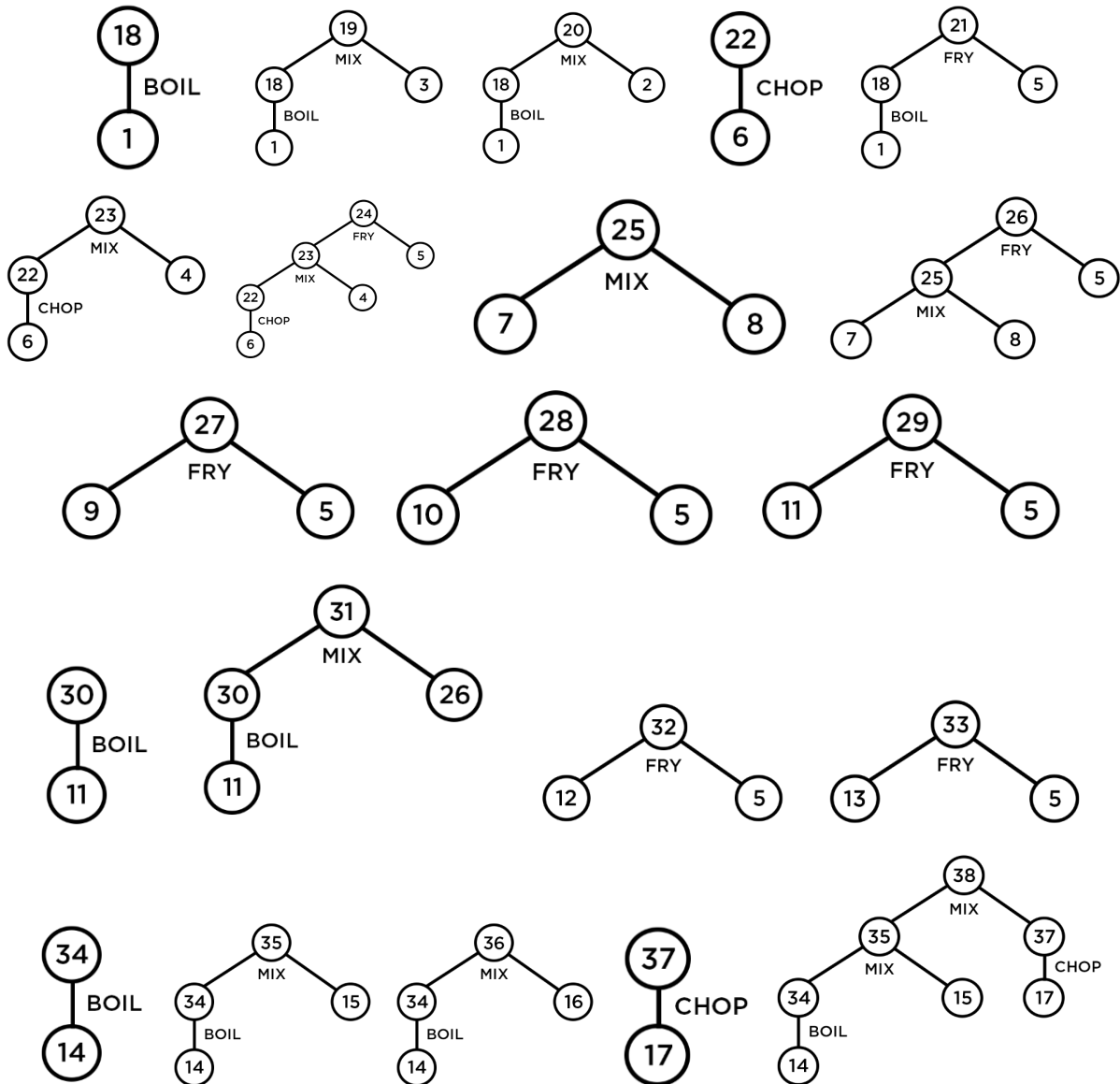
6.4 Dataset Resep Makanan

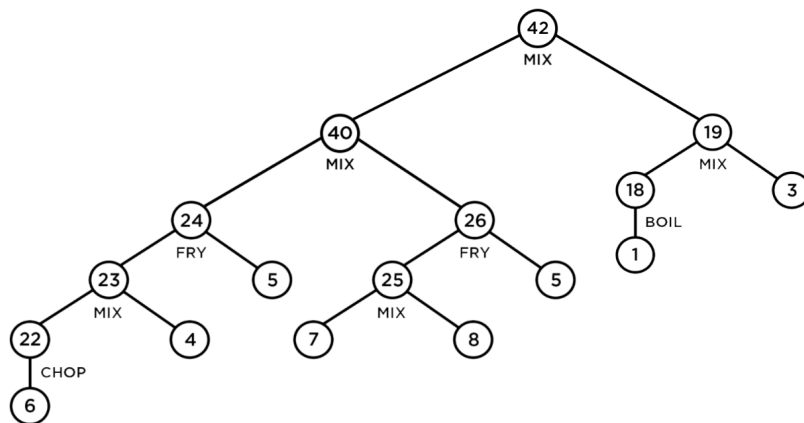
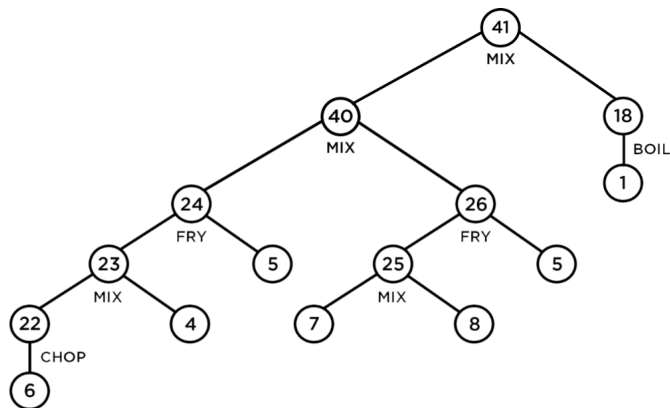
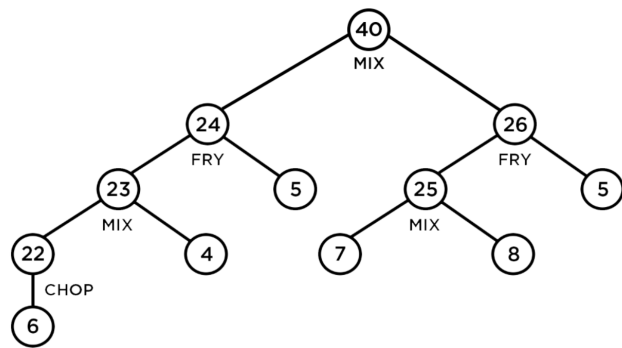
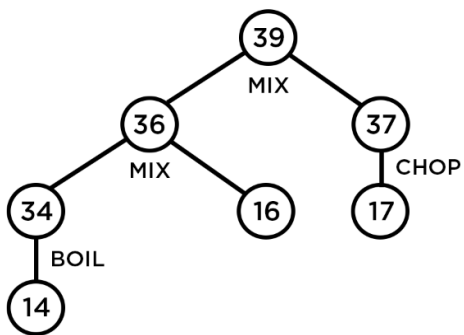
Pada dataset resep makanan, program akan menerima dataset resep makanan yang telah dibuat terlebih dahulu sebelumnya dalam file .txt dengan baris pertama adalah iterasi untuk jumlah total resep makanan yang ada dalam program. Berikutnya, untuk setiap resep makanan, terdapat nomor ID yang merupakan parent dari makanan, lalu ada banyaknya child dari makanan dengan ID yang dituliskan sebelumnya dan juga ada semua ID makanan yang merupakan child dari parent makanannya. Dataset yang ada ini kemudian akan dibaca dan digunakan untuk membangun tree proses pengolahan makanan.

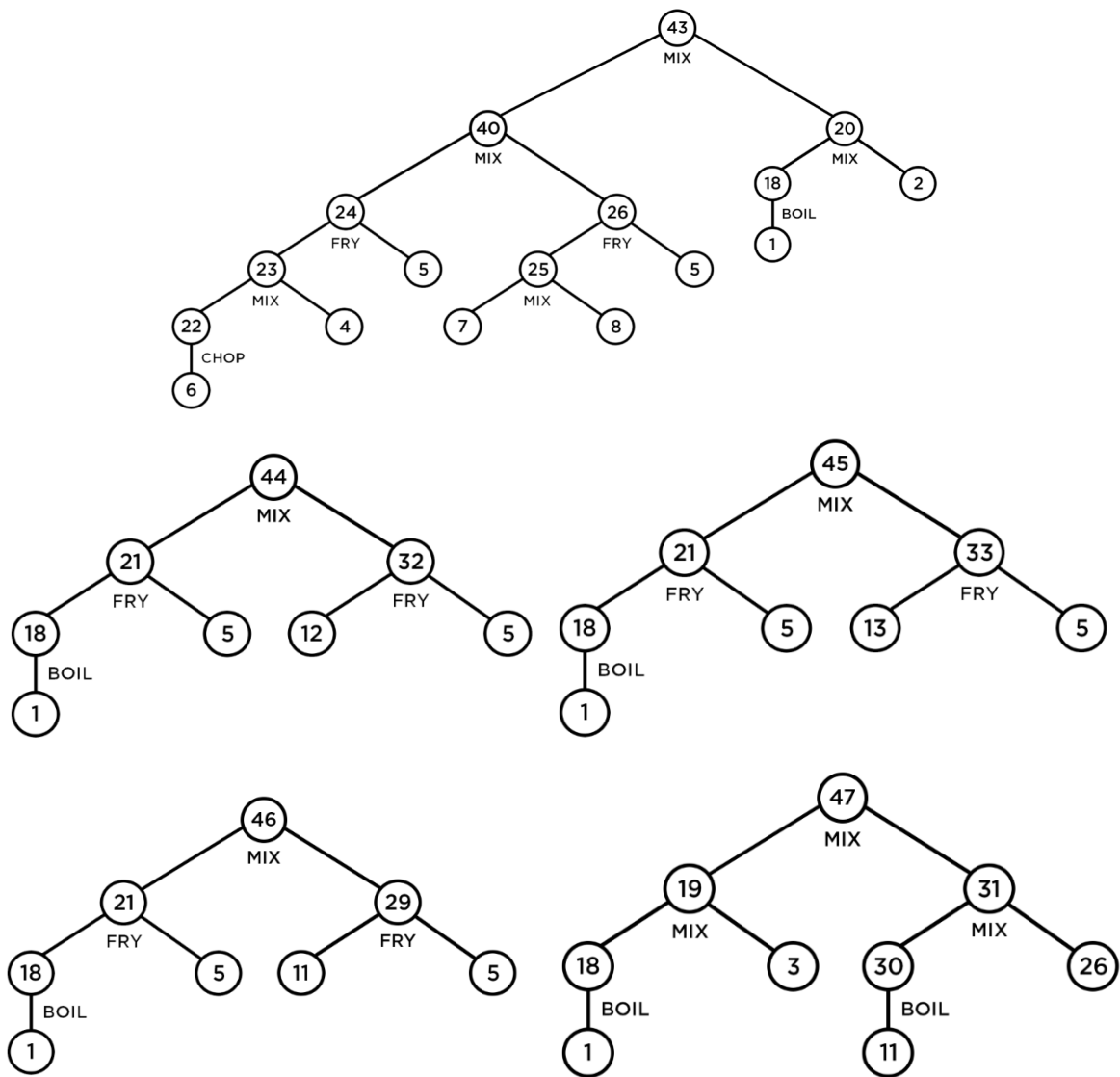
1	30	9	25 2 7 8	17	33 2 13 5	25	41 2 40 18
2	18 1 1	10	26 2 25 5	18	34 1 14	26	42 2 40 19
3	19 2 18 3	11	27 2 9 5	19	35 2 34 15	27	43 2 40 20
4	20 2 18 2	12	28 2 10 5	20	36 2 34 16	28	44 2 32 21
5	21 2 18 5	13	29 2 11 5	21	37 1 17	29	45 2 33 21
6	22 1 6	14	30 1 11	22	38 2 37 35	30	46 2 29 21
7	23 2 22 4	15	31 2 30 26	23	39 2 37 36	31	47 2 31 19.
8	24 2 23 5	16	32 2 12 5	24	40 2 26 24		

Gambar 6.4.1 Tampilan File .txt untuk dataset resep makanan

Tree untuk Dataset Resep Makanan:







Gambar 6.4.2 Gambar pemodelan tree untuk dataset resep makanan

6.5 Command Inisiasi (START dan EXIT)

6.5.1 Start

Pada command START, program akan menerima input “START”. Setelah input dimasukkan, program utama akan langsung otomatis dimulai.

```
Welcome to Diner Dash CLI Edition!  
Please enter your name: Player Test  
Please write START to start the game or EXIT to exit!  
>> START
```

Gambar 6.5.1.1 Tampilan Command START pada program

6.5.2 Exit

Pada command EXIT, program akan menerima input “EXIT”. Setelah input dimasukkan, program akan langsung berhenti dan menampilkan splash screen yang menunjukkan bahwa program telah berhenti dijalankan.



Gambar 6.5.2.1 Tampilan Command EXIT pada program

6.6 Command Pemesanan (BUY dan DELIVERY)

6.6.1 Buy

Pada command BUY, program akan menerima input “BUY”. Setelah input dimasukkan, program akan mengecek kondisi posisi dari simulator BNMO. Jika posisi simulator tidak dekat dengan titik lokasi fitur BUY (pada peta ditandai dengan B), program akan menampilkan pesan bahwa simulator tidak berada di toko dan tidak dapat untuk melakukan pembelian. Jika posisi simulator sudah dekat dengan titik lokasi fitur BUY, program akan menampilkan list dari bahan makanan yang dapat dibeli oleh simulator BNMO dan meminta input berupa ID makanan yang ingin dibeli. Jika ID makanan tidak ada dalam dataset, program akan menampilkan “ID tidak ditemukan.” Jika ID makanan valid, program akan menampilkan pesan bahwa makanan sudah berhasil dibeli dan tinggal ditunggu untuk sampai. Setelah makanan sampai, maka akan muncul notifikasi bahwa makanan sudah berhasil dibeli dan menambahkan makanan yang dibeli tersebut pada inventory.

```
=====
EZ di posisi: (1,5)
Waktu saat ini: 00:00:10
Notifikasi: -
*****
*                               *
*      S T   X   X   X   *
*              X X X   X   *
*              X   X   X   *
*      M                               *
*      X                               *
*      X                               *
*      X X X   F                               *
*      *                               *
*      B               X X X X   *
*              X                               *
*              X X X   *
*      K               X                               *
*              X X X X   *
*      *                               *
*****
Masukkan command: BUY
=====
      BUY      =
=====
List Bahan Makanan:
=====
=====DAFTAR MAKANAN=====
ID makanan:      1
Nama makanan:    Beras
Expired:          30:00:00
Delivery time:    00:00:20
Lokasi aksi:      Buy
=====
ID makanan:      2
Nama makanan:    Santan
Expired:          12:00:00
Delivery time:    00:00:10
Lokasi aksi:      Buy
=====
ID makanan:      3
Nama makanan:    Kunyit
Expired:          17:15:00
Delivery time:    00:00:05
Lokasi aksi:      Buy
=====
ID makanan:      4
Nama makanan:    Tepung
Expired:          25:10:00
Delivery time:    00:00:10
Lokasi aksi:      Buy
=====
ID makanan:      5
Nama makanan:    Minyak Goreng
Expired:          10:00:00
Delivery time:    00:00:15
Lokasi aksi:      Buy
=====
ID makanan:      6
Nama makanan:    Ayam Mentah
Expired:          22:00:00
Delivery time:    00:00:30
Lokasi aksi:      Buy
=====
ID makanan:      7
Nama makanan:    Cabai
Expired:          03:00:00
Delivery time:    00:00:05
Lokasi aksi:      Buy
=====
ID makanan:      8
Nama makanan:    Bawang
Expired:          00:12:00
Delivery time:    00:00:05
Lokasi aksi:      Buy
=====
ID makanan:      9
Nama makanan:    Tahu Mentah
Expired:          05:10:00
Delivery time:    00:00:20
Lokasi aksi:      Buy
=====
ID makanan:      10
Nama makanan:    Tempe Mentah
Expired:          05:10:00
Delivery time:    00:00:20
Lokasi aksi:      Buy
=====
ID makanan:      11
Nama makanan:    Telur
Expired:          05:10:00
Delivery time:    00:00:05
Lokasi aksi:      Buy
=====
ID makanan:      12
Nama makanan:    Bakso Mentah
Expired:          08:20:00
Delivery time:    00:00:25
Lokasi aksi:      Buy
=====
ID makanan:      13
Nama makanan:    Sosis Mentah
Expired:          08:20:00
Delivery time:    00:00:25
Lokasi aksi:      Buy
=====
ID makanan:      14
Nama makanan:    Air
Expired:          50:00:00
Delivery time:    00:00:05
Lokasi aksi:      Buy
=====
ID makanan:      15
Nama makanan:    Teh Sachet
Expired:          35:00:00
Delivery time:    00:00:05
Lokasi aksi:      Buy
=====
ID makanan:      16
Nama makanan:    Kopi Sachet
Expired:          35:00:00
Delivery time:    00:00:05
Lokasi aksi:      Buy
=====
ID makanan:      17
Nama makanan:    Es Batu Bongkahan
Expired:          00:00:30
Delivery time:    00:00:05
Lokasi aksi:      Buy
=====
```

Gambar 6.6.1.1 Tampilan saat menjalankan fitur BUY ketika simulator berada di dekat T (toko)

```

=====
EZ di posisi: (0,0)
Waktu saat ini: 00:00:00
* * * * *
* S *
*      T   X X X *
*      X X X X *
*      X X X *
*      M *
*      X *
*      X   C *
*      X X X F *
*      *
*      B   X X X X *
*      X *
*      X X X *
*      K   X *
*      X X X X *
*      *
* * * * *
=====
Masukkan command: BUY
Anda tidak berada di toko!

```

Gambar 6.6.1.2 Tampilan saat menjalankan fitur BUY ketika simulator tidak berada di dekat T (toko)

```

=====
Masukkan command: buy
== Masukan Tidak Valid ==
=====
EZ di posisi: (1,5)
Waktu saat ini: 00:00:11
Notifikasi: -
* * * * *
*      S T   X X X *
*      X X X X *
*      X X X *
*      M *
*      X *
*      X   C *
*      X X X F *
*      *
*      B   X X X X *
*      X *
*      X X X *
*      K   X *
*      X X X X *
*      *
* * * * *
=====
Masukkan command: 

```

Gambar 6.6.1.3 Tampilan saat menjalankan fitur BUY ketika simulator berada di dekat T (toko) dengan command yang salah

```

Masukkan ID makanan yang ingin dibeli: 18
ID tidak ditemukan
=====
E for EZ di posisi: (1,5)
Waktu saat ini: 00:00:10
Notifikasi: -
* * * * *
*           S T   X X X *
*           X X X X *
*           X X X X *
*           *
*   X   M           *
*   X           C   *
*   X X X   F           *
*           *
*   B           X X X X *
*           X           *
*           X X X       *
*           X           *
*           X X X X     *
*           *
* * * * *
Masukkan command:

```

Gambar 6.6.1.4 Tampilan saat memasukkan ID makanan yang ingin dibeli tetapi ID tidak terdaftar dalam daftar menu.

```

Masukkan ID makanan yang ingin dibeli: 3
Makanan berhasil dicheckout!
Silahkan menunggu makanan sampai di lokasi Anda!
=====
EZ di posisi: (1,5)
Waktu saat ini: 00:00:11
Notifikasi: -
* * * * *
*           S T   X X X *
*           X X X X *
*           X X X X *
*           *
*   X   M           *
*   X           C   *
*   X X X   F           *
*           *
*   B           X X X X *
*           X           *
*           X X X       *
*           X           *
*           X X X X     *
*           *
* * * * *
=====
Masukkan command:

```

Gambar 6.6.1.5 Tampilan saat memasukkan ID makanan yang ingin dibeli dan valid

6.6.2 Delivery

Pada command DELIVERY, program akan menerima input “DELIVERY”. Setelah input dimasukkan, program akan menampilkan list makanan apa saja yang sedang dalam proses diantar. Jika tidak ada makanan yang sedang dalam proses diantar, program akan menampilkan “tidak ada bahan makanan yang sedang diantar.”

```
=====
Masukkan command: DELIVERY
Daftar barang yang sedang diantar:
    Tidak ada barang yang sedang diantar
=====
```

Gambar 6.6.2.1 Tampilan saat menjalankan fitur DELIVERY saat belum ada makanan yang diantar

```
=====
Masukkan command: DELIVERY
Daftar barang yang sedang diantar:
    1. Kunyit - 4 menit
=====
```

Gambar 6.6.2.2 Tampilan saat menjalankan fitur DELIVERY saat setelah membeli makanan

6.7 Command Peta (MOVE NORTH, MOVE EAST, MOVE WEST, MOVE SOUTH)

6.7.1 Move North

Pada command MOVE NORTH, program akan menerima input “MOVE NORTH”. Setelah input dimasukkan, program akan mengecek kondisi posisi dari simulator BNMO. Jika posisi simulator sudah berada pada ujung atas matriks ataupun sudah terhalang oleh lokasi fitur,

maka program akan menampilkan pesan bahwa command tidak valid. Jika tidak, program akan mengubah posisi simulator BNMO sebanyak satu posisi ke arah utara (atas) dari petak matriks posisi sebelumnya. Selain itu, waktu juga akan mengalami perubahan yaitu akan bertambah sebanyak 1 menit.

```

=====
E for EZ di posisi: (0,0)
Waktu saat ini: 00:00:00
* * * * *
* S *
*   T   X X X *
*   X X X *
*   X X X *
*   M   *
* X     *
* X     *
* X X X   F   C   *
*   B       X X X X *
*           X   *
*           X X X *
*       K     X   *
*           X X X X *
* * * * *
=====
Masukkan command: MOVE NORTH
Gagal berpindah
=====
E for EZ di posisi: (0,0)
Waktu saat ini: 00:00:00
Notifikasi: -
* * * * *
* S *
*   T   X X X *
*   X X X *
*   X X X *
*   M   *
* X     *
* X     *
* X X X   F   C   *
*   B       X X X X *
*           X   *
*           X X X *
*       K     X   *
*           X X X X *
* * * * *
=====
Masukkan command: 

```

Gambar 6.7.1.1 Tampilan saat memasukkan command MOVE NORTH tetapi kondisi tidak valid untuk bergerak (berlaku untuk semua arah lain)

```

E for EZ di posisi: (1,0)
Waktu saat ini: 00:00:03
Notifikasi: -
*****
* S      T  X  X  X *
*      X X X  X *
*      X  X  X *
*    M      *
*  X      *
*  X      C      *
* X X X  F      *
*      *
*    B      X X X X *
*      X      *
*      X X X      *
*      K      X      *
*      X X X X      *
*      *
*****

Masukkan command: MOVE NORTH

=====
E for EZ di posisi: (0,0)
Waktu saat ini: 00:00:04
Notifikasi: -
*****
* S      T  X  X  X *
*      X X X  X *
*      X  X  X *
*    M      *
*  X      *
*  X      C      *
* X X X  F      *
*      *
*    B      X X X X *
*      X      *
*      X X X      *
*      K      X      *
*      X X X X      *
*      *
*****

Masukkan command:

```

Gambar 6.7.1.2 Tampilan saat memasukkan command MOVE NORTH dan kondisi valid

6.7.2 Move East

Pada command MOVE EAST, program akan menerima input “MOVE EAST”. Setelah input dimasukkan, program akan mengecek kondisi posisi dari simulator BNMO. Jika posisi simulator sudah berada pada ujung kanan matriks ataupun sudah terhalang oleh lokasi fitur, maka program akan menampilkan pesan bahwa command tidak valid. Jika tidak, program akan mengubah posisi simulator BNMO sebanyak satu posisi ke arah timur (kanan) dari petak matriks posisi sebelumnya. Selain itu, waktu juga akan mengalami perubahan yaitu akan bertambah sebanyak 1 menit.

```

Masukkan command: MOVE NORTH
Gagal berpindah
=====
E for EZ di posisi: (0,0)
Waktu saat ini: 00:00:00
Notifikasi: -
*****
* S                               *
*      T      X  X  X  *
*      X X X  X  *
*      X  X  X  *
*      M                               *
*      X                               *
*      X                               *
*      X X X      F      C      *
*                               *
*      B      X X X X  *
*      X                               *
*      X X X  *
*      K      X      *
*      X X X X  *
*                               *
*****

Masukkan command: MOVE EAST
=====
E for EZ di posisi: (0,1)
Waktu saat ini: 00:00:01
Notifikasi: -
*****
* S                               *
*      T      X  X  X  *
*      X X X  X  *
*      X  X  X  *
*      M                               *
*      X                               *
*      X                               *
*      X X X      F      C      *
*                               *
*      B      X X X X  *
*      X                               *
*      X X X  *
*      K      X      *
*      X X X X  *
*                               *
*****
Masukkan command: 

```

Gambar 6.7.2.1 Tampilan saat memasukkan command MOVE EAST dan kondisi valid

6.7.3 Move West

Pada command MOVE WEST, program akan menerima input “MOVE WEST”. Setelah input dimasukkan, program akan mengecek kondisi posisi dari simulator BNMO. Jika posisi simulator sudah berada pada ujung kiri matriks ataupun sudah terhalang oleh lokasi fitur, maka program akan menampilkan pesan bahwa command tidak valid. Jika tidak, program akan mengubah posisi simulator BNMO sebanyak satu posisi ke arah barat (kiri) dari petak matriks posisi sebelumnya. Selain itu, waktu juga akan mengalami perubahan yaitu akan bertambah sebanyak 1 menit.

```

E for EZ di posisi: (1,1)
Waktu saat ini: 00:00:02
Notifikasi: -
*****
* S      T  X X X *
*      X X X X *
*      X X X *
*   M      *
* X      *
* X      C *
* X X X  F *
*      *
* B      X X X X *
*      X *
*      X X X *
*      K  X *
*      X X X X *
*      *
*****

=====
Masukkan command: MOVE WEST
=====

E for EZ di posisi: (1,0)
Waktu saat ini: 00:00:03
Notifikasi: -
*****
* S      T  X X X *
*      X X X X *
*      X X X *
*   M      *
* X      *
* X      C *
* X X X  F *
*      *
* B      X X X X *
*      X *
*      X X X *
*      K  X *
*      X X X X *
*      *
*****

Masukkan command:

```

Gambar 6.7.3.1 Tampilan saat memasukkan command MOVE WEST dan kondisi valid

6.7.4 Move South

Pada command MOVE SOUTH, program akan menerima input “MOVE SOUTH”. Setelah input dimasukkan, program akan mengecek kondisi posisi dari simulator BNMO. Jika posisi simulator sudah berada pada ujung bawah matriks ataupun sudah terhalang oleh lokasi fitur, maka program akan menampilkan pesan bahwa command tidak valid. Jika tidak, program akan mengubah posisi simulator BNMO sebanyak satu posisi ke arah selatan (bawah) dari petak matriks posisi sebelumnya. Selain itu, waktu juga akan mengalami perubahan yaitu akan bertambah sebanyak 1 menit.

```

E for EZ di posisi: (0,1)
Waktu saat ini: 00:00:01
Notifikasi: -
*****
* S          *
*      T    X X X *
*      X X X X *
*      X X X *
*      M          *
*      X          *
*      X          C *
*      X X X    F *
*      B          *
*      X X X X *
*      X          *
*      X X X *
*      K      X *
*      X X X X *
*****

=====
Masukkan command: MOVE SOUTH
=====

E for EZ di posisi: (1,1)
Waktu saat ini: 00:00:02
Notifikasi: -
*****
* S          *
*      T    X X X *
*      X X X X *
*      X X X *
*      M          *
*      X          *
*      X          C *
*      X X X    F *
*      B          *
*      X X X X *
*      X          *
*      X X X *
*      K      X *
*      X X X X *
*****

=====
Masukkan command: 

```

Gambar 6.7.4.1 Tampilan saat memasukkan command MOVE SOUTH dan kondisi valid

6.8 Command Pengolahan (MIX, CHOP, FRY, BOIL)

6.8.1 Mix

Pada command MIX, program akan menerima input “MIX”. Setelah input dimasukkan, program akan mengecek kondisi posisi dari simulator BNMO. Jika posisi simulator tidak dekat dengan titik lokasi fitur MIXER (pada peta ditandai dengan M), program akan menampilkan pesan bahwa simulator BNMO tidak berada didekat MIXER dan tidak dapat untuk melakukan proses MIX.

```

Masukkan command: MIX
Anda tidak berada di dekat MIXER.
=====
E for EZ di posisi: (0,0)
Waktu saat ini: 00:00:00
Notifikasi: -
*****
* S                                     *
*      T   X   X   X   *
*           X X X   X   *
*           X   X   X   *
*      M                                     *
*   X                                     *
*   X                                     *
* X X X   F   C                                     *
*      B           X X X X   *
*           X                                     *
*           X X X   *
*      K           X                                     *
*           X X X X   *
*****

```

Gambar 6.8.1.1 Menjalankan fitur MIX saat simulator tidak berada di dekat MIXER

Jika posisi simulator BNMO sudah dekat dengan titik lokasi fitur MIX, program akan menampilkan berbagai macam list makanan yang dapat dibuat dengan cara MIX.

```

Masukkan command: MIX
=====
=      MIX      =
=====
List Bahan Makanan yang Bisa Dibuat:
  1. Nasi Kuning
  2. Nasi Uduk
  3. Ayam Tepung
  4. Sambalhan
  5. Telur Balado
  6. Teh Panas
  7. Kopi Panas
  8. Es Tehhan
  9. Kopi Esan
 10. Ayam Goreng Extra Sambal
 11. Nasi Putih dengan Ayam Goreng Extra Sambal
 12. Nasi Kuning dengan Ayam Goreng Extra Sambal
 13. Nasi Uduk dengan Ayam Goreng Extra Sambal
 14. Nasi Goreng dengan Bakso
 15. Nasi Goreng dengan Sosis
 16. Nasi Goreng dengan Telur Dadar
 17. Nasi Kuning dengan Telur Balado

Kirim 0 untuk exit.

```

Gambar 6.8.1.2 Menjalankan fitur MIX saat simulator berada di dekat MIXER

Program juga akan meminta input angka untuk memilih makanan yang ingin dibuat. Jika angka input adalah 0, program akan langsung keluar dari menu MIX dan kembali ke program utama.

```

List Bahan Makanan yang Bisa Dibuat:
1. Nasi Kuning
2. Nasi Uduk
3. Ayam Tepung
4. Sambalhan
5. Telur Balado
6. Teh Panas
7. Kopi Panas
8. Es Tehhan
9. Kopi Esan
10. Ayam Goreng Extra Sambal
11. Nasi Putih dengan Ayam Goreng Extra Sambal
12. Nasi Kuning dengan Ayam Goreng Extra Sambal
13. Nasi Uduk dengan Ayam Goreng Extra Sambal
14. Nasi Goreng dengan Bakso
15. Nasi Goreng dengan Sosis
16. Nasi Goreng dengan Telur Dadar
17. Nasi Kuning dengan Telur Balado

Kirim 0 untuk exit.
Enter Command: 0

=====
E for EZ di posisi: (3,3)
Waktu saat ini: 00:00:13
Notifikasi: -
*****
*               *
*           T   X X X *
*           X X X X *
*       S       X X X *
*       M               *
*   X               *
*   X               C   *
*   X X X   F               *
*       B               X X X X *
*               X         *
*               X X X     *
*       K               X         *
*               X X X X   *
*               *
*****

```

Gambar 6.8.1.3 Menjalankan fitur MIX yang valid dan memilih angka 0

Jika angka input tidak ada dalam list, maka program akan melakukan loop untuk input angka hingga valid.

```

List Bahan Makanan yang Bisa Dibuat:
1. Nasi Kuning
2. Nasi Uduk
3. Ayam Tepung
4. Sambalhan
5. Telur Balado
6. Teh Panas
7. Kopi Panas
8. Es Tehhan
9. Kopi Esan
10. Ayam Goreng Extra Sambal
11. Nasi Putih dengan Ayam Goreng Extra Sambal
12. Nasi Kuning dengan Ayam Goreng Extra Sambal
13. Nasi Uduk dengan Ayam Goreng Extra Sambal
14. Nasi Goreng dengan Bakso
15. Nasi Goreng dengan Sosis
16. Nasi Goreng dengan Telur Dadar
17. Nasi Kuning dengan Telur Balado

Kirim 0 untuk exit.
Enter Command: 18
Input tidak valid. Silakan masukkan input kembali:

```

Gambar 6.8.1.4 Menjalankan fitur MIX yang valid dan memilih angka yang tidak terdapat di list makanan yang dapat di-mix

Jika input angka ada, berikutnya program akan melakukan pengecekan pada inventory untuk mengecek ketersediaan bahan untuk membuat makanan. Jika bahan makanan yang diperlukan tidak tersedia dalam inventory, program akan menampilkan pesan gagal membuat makanan karena bahan makanannya tidak ada.

```

Enter Command: 1
Gagal membuat Nasi Kuning karena kamu tidak memiliki bahan berikut:
1. Nasi Putih
2. Kunyit
=====
E for EZ di posisi: (3,3)
Waktu saat ini: 00:00:11
Notifikasi: -
* * * * *
*           T   X   X   X   *
*           XXXX   *
*           X   X   X   *
*       S           *
*       M           *
*   X           *
*   X           *
*   XXX   F           *
*           B           XXXXX *
*           K           XXXX  *
*           XXXXX *
* * * * *
=====

```

Gambar 6.8.1.5 Menjalankan fitur MIX yang valid dan memilih salah satu makanan saat inventory tidak ada bahan yang dibutuhkan

Jika bahan makanan yang diperlukan ada, maka program akan menampilkan pesan bahwa makanan sudah berhasil dibuat dan sudah dimasukkan dalam inventory serta langsung menghapus bahan makanan yang digunakan pada list inventory.

```

Kirim 0 untuk exit.
Enter Command: 1
Nasi Kuning selesai dibuat dan sudah masuk ke inventory!
=====
E for EZ di posisi: (3,3)
Waktu saat ini: 00:03:37
Notifikasi:
1. Nasi Kuning sudah diterima oleh BNMO
* * * * *
*           T   X   X   X   *
*           XXXX   *
*           X   X   X   *
*       S           *
*       M           *
*   X           *
*   X           *
*   XXX   F           *
*           B           XXXXX *
*           K           XXXX  *
*           XXXXX *
* * * * *
=====

```

Gambar 6.8.1.6 Menjalankan fitur MIX yang valid dan inventory terdapat bahan makanan yang diperlukan untuk di-mix

6.8.2 Chop

Pada command CHOP, program akan menerima input “CHOP”. Setelah input dimasukkan, program akan mengecek kondisi posisi dari simulator BNMO. Jika posisi simulator tidak dekat dengan titik lokasi fitur CHOPPER (pada peta ditandai dengan C), program akan menampilkan pesan bahwa simulator BNMO tidak berada didekat CHOPPER dan tidak dapat untuk melakukan proses CHOP.

```
=====
Masukkan command: CHOP
Anda tidak berada di dekat CHOPPER.
=====
E for EZ di posisi: (6,9)
Waktu saat ini: 00:01:10
Notifikasi: -
* * * * *
*           T   X X X *
*          XXXX *
*          XX *
*         M *
*        X *
*       X *
*      XXX *
*     B *
*          K *
*          XXXX *
*          X *
*          XXX *
*          X *
*          XXXX *
* * * * *
```

Gambar 6.8.2.1 Menjalankan fitur CHOP saat simulator tidak berada di dekat CHOPPER

Jika posisi simulator BNMO sudah dekat dengan titik lokasi fitur CHOP, program akan menampilkan berbagai macam list makanan yang dapat dibuat dengan cara CHOP. Program juga akan meminta input angka untuk memilih makanan yang ingin dibuat.


```
=====
=      CHOP      =
=====
List Bahan Makanan yang Bisa Dibuat:
  1. Ayam Potong
  2. Es Batuan

Kirim 0 untuk exit.
Enter Command: 3
Input tidak valid. Silakan masukkan input kembali: |
```

Gambar 6.8.2.4 Menjalankan fitur CHOP yang valid dan memilih angka yang tidak terdapat di list makanan yang dapat di-chop

Jika input angka ada, berikutnya program akan melakukan pengecekan pada inventory untuk mengecek ketersediaan bahan untuk pemotongan makanan. Jika bahan makanan yang diperlukan tidak tersedia dalam inventory, program akan menampilkan pesan gagal memotong makanan karena bahan makanannya tidak ada.

```
=====
=      CHOP      =
=====
List Bahan Makanan yang Bisa Dibuat:
  1. Ayam Potong
  2. Es Batuan

Kirim 0 untuk exit.
Enter Command: 1
Gagal membuat Ayam Potong karena kamu tidak memiliki bahan berikut:
  1. Ayam Mentah
=====
```

Gambar 6.8.2.5 Menjalankan fitur CHOP yang valid dan memilih salah satu makanan saat inventory tidak ada bahan yang dibutuhkan

Jika bahan makanan yang diperlukan ada, maka program akan menampilkan pesan bahwa makanan sudah berhasil dipotong dan sudah dimasukkan dalam inventory serta langsung menghapus bahan makanan yang digunakan pada list inventory.

```
=====
=      CHOP      =
=====
List Bahan Makanan yang Bisa Dibuat:
  1. Ayam Potong
  2. Es Batuan

Kirim 0 untuk exit.
Enter Command: 1
Ayam Potong selesai dibuat dan sudah masuk ke inventory!
=====
```

Gambar 6.8.2.6 Menjalankan fitur CHOP yang valid dan inventory terdapat bahan makanan yang diperlukan untuk di-chop

6.8.3 Fry

Pada command FRY, program akan menerima input “FRY”. Setelah input dimasukkan, program akan mengecek kondisi posisi dari simulator BNMO. Jika posisi simulator tidak dekat dengan titik lokasi fitur FRYER (pada peta ditandai dengan F), program akan menampilkan pesan bahwa simulator BNMO tidak berada didekat CHOPPER dan tidak dapat untuk melakukan proses FRY.

```
Masukkan command: FRY
Anda tidak berada di dekat FRYER.
=====
E for EZ di posisi: (6,9)
Waktu saat ini: 00:01:10
Notifikasi: -
*****
*               T   X X X   *
*               XXXX X   *
*               X X X   *
*           M       *
*   X             *
*   X             S C   *
*   XXX F         *
*   B             XXXXX  *
*               X       *
*               XXXX    *
*           K       X    *
*               XXXXX   *
*****
```

Gambar 6.8.3.1 Menjalankan fitur FRY saat simulator tidak berada di dekat FRYER

Jika posisi simulator BNMO sudah dekat dengan titik lokasi fitur FRY, program akan menampilkan berbagai macam list makanan yang dapat dibuat dengan cara FRY.

```
Masukkan command: FRY
=====
=       FRY       =
=====
List Bahan Makanan yang Bisa Dibuat:
1. Nasi Goreng
2. Ayam Goreng
3. Sambal Goreng
4. Tahu Goreng
5. Tempe Goreng
6. Telur Dadar
7. Bakso Goreng
8. Sosis Goreng

Kirim 0 untuk exit.
Enter Command: █
```

Gambar 6.8.3.2 Menjalankan fitur FRY saat simulator berada di dekat FRYER

Program juga akan meminta input angka untuk memilih makanan yang ingin dibuat. Jika angka input adalah 0, program akan langsung keluar dari menu FRY dan kembali ke program utama.

```
=====
=       FRY       =
=====
List Bahan Makanan yang Bisa Dibuat:
1. Nasi Goreng
2. Ayam Goreng
3. Sambal Goreng
4. Tahu Goreng
5. Tempe Goreng
6. Telur Dadar
7. Bakso Goreng
8. Sosis Goreng

Kirim 0 untuk exit.
Enter Command: 0

=====
E for EZ di posisi: (7,5)
Waktu saat ini: 00:01:23
Notifikasi: -
*****
*               *
*       T       *
*       X X X   *
*       X X X   *
*       X X X   *
*       M       *
*       X       *
*       X       *
*       X X X   *
*       S F     *
*       B       *
*       X X X X  *
*       X       *
*       X X X   *
*       K       *
*       X X X   *
*       X X X   *
*       *       *
*****
```

Gambar 6.8.3.3 Menjalankan fitur FRY yang valid dan memilih angka 0

Jika angka input tidak ada dalam list, maka program akan melakukan loop untuk input angka hingga valid.

```
List Bahan Makanan yang Bisa Dibuat:
1. Nasi Goreng
2. Ayam Goreng
3. Sambal Goreng
4. Tahu Goreng
5. Tempe Goreng
6. Telur Dadar
7. Bakso Goreng
8. Sosis Goreng

Kirim 0 untuk exit.
Enter Command: 10
Input tidak valid. Silakan masukkan input kembali: 
```

Gambar 6.8.3.4 Menjalankan fitur FRY yang valid dan memilih angka yang tidak terdapat di list makanan yang dapat di-fry

Jika input angka ada, berikutnya program akan melakukan pengecekan pada inventory untuk mengecek ketersediaan bahan untuk penggorengan makanan. Jika bahan makanan yang diperlukan tidak tersedia dalam inventory, program akan menampilkan pesan gagal menggoreng makanan karena bahan makanannya tidak ada.

```
=====
=      FRY      =
=====
List Bahan Makanan yang Bisa Dibuat:
1. Nasi Goreng
2. Ayam Goreng
3. Sambal Goreng
4. Tahu Goreng
5. Tempe Goreng
6. Telur Dadar
7. Bakso Goreng
8. Sosis Goreng

Kirim 0 untuk exit.
Enter Command: 1
Gagal membuat Nasi Goreng karena kamu tidak memiliki bahan berikut:
1. Nasi Putih
2. Minyak Goreng
=====
```

Gambar 6.8.3.5 Menjalankan fitur FRY yang valid dan memilih salah satu makanan saat inventory tidak ada bahan yang dibutuhkan

Jika bahan makanan yang diperlukan ada, maka program akan menampilkan pesan bahwa makanan sudah berhasil digoreng dan sudah dimasukkan dalam inventory serta langsung menghapus bahan makanan yang digunakan pada list inventory.

```
Masukkan command: FRY
=====
=      FRY      =
=====
List Bahan Makanan yang Bisa Dibuat:
1. Nasi Goreng
2. Ayam Goreng
3. Sambal Goreng
4. Tahu Goreng
5. Tempe Goreng
6. Telur Dadar
7. Bakso Goreng
8. Sosis Goreng

Kirim 0 untuk exit.
Enter Command: 1
Nasi Goreng selesai dibuat dan sudah masuk ke inventory!
=====
E for EZ di posisi: (7,5)
Waktu saat ini: 00:03:46
Notifikasi:
1. Nasi Goreng sudah diterima oleh BMMO
=====
```

Gambar 6.8.3.6 Menjalankan fitur FRY yang valid dan inventory terdapat bahan makanan yang diperlukan untuk di-fry

6.8.4 Boil

Pada command BOIL, program akan menerima input “BOIL”. Setelah input dimasukkan, program akan mengecek kondisi posisi dari simulator BNMO. Jika posisi simulator tidak dekat dengan titik lokasi fitur BOILER (pada peta ditandai dengan B), program akan menampilkan pesan bahwa simulator BNMO tidak berada didekat BOILER dan tidak dapat untuk melakukan proses BOIL.

```
Masukkan command: BOIL
Anda tidak berada di dekat BOILER.
=====
E for EZ di posisi: (6,9)
Waktu saat ini: 00:01:10
Notifikasi: -
* * * * *
*           T   X X X *
*           X X X X *
*           X X X *
*   M       *
* X         *
* X         S C *
* X X X   F *
*   B       X X X X *
*           X *
*           X X X *
*           X *
*           X X X X *
* * * * *
```

Gambar 6.8.4.1 Menjalankan fitur BOIL saat simulator tidak berada di dekat BOILER

Jika posisi simulator BNMO sudah dekat dengan titik lokasi fitur BOIL, program akan menampilkan berbagai macam list makanan yang dapat dibuat dengan cara BOIL.

```
Masukkan command: BOIL
=====
=      BOIL      =
=====
List Bahan Makanan yang Bisa Dibuat:
1. Nasi Putih
2. Telur Rebus
3. Air Panas

Kirim 0 untuk exit.
Enter Command: 
```

Gambar 6.8.4.2 Menjalankan fitur BOIL saat simulator berada di dekat BOILER

Program juga akan meminta input angka untuk memilih makanan yang ingin dibuat. Jika angka input adalah 0, program akan langsung keluar dari menu BOIL dan kembali ke program utama.

```
=====
=      BOIL      =
=====
List Bahan Makanan yang Bisa Dibuat:
  1. Nasi Putih
  2. Telur Rebus
  3. Air Panas

Kirim 0 untuk exit.
Enter Command: 0
=====
E for EZ di posisi: (9,1)
Waktu saat ini: 00:01:47
Notifikasi: -
*****
*               *
*      T   X X X *
*      X X X X *
*      X X X *
*      M           *
*      X           *
*      X           *
*      X X X   F   C *
*               *
*      S B           X X X X *
*               X *
*               X X X *
*      K           X *
*               X X X X *
*               *
*****
```

Gambar 6.8.4.3 Menjalankan fitur BOIL yang valid dan memilih angka 0

Jika angka input tidak ada dalam list, maka program akan melakukan loop untuk input angka hingga valid.

```
=====
=      BOIL      =
=====
List Bahan Makanan yang Bisa Dibuat:
  1. Nasi Putih
  2. Telur Rebus
  3. Air Panas

Kirim 0 untuk exit.
Enter Command: 19
Input tidak valid. Silakan masukkan input kembali: 
```

Gambar 6.8.4.4 Menjalankan fitur BOIL yang valid dan memilih angka yang tidak terdapat di list makanan yang dapat di-boil

Jika input angka ada, berikutnya program akan melakukan pengecekan pada inventory untuk mengecek ketersediaan bahan untuk perebusan makanan. Jika bahan makanan yang

diperlukan tidak tersedia dalam inventory, program akan menampilkan pesan gagal merebus makanan karena bahan makanannya tidak ada.

```
=====
=      BOIL      =
=====
List Bahan Makanan yang Bisa Dibuat:
  1. Nasi Putih
  2. Telur Rebus
  3. Air Panas

Kirim 0 untuk exit.
Enter Command: 1
Gagal membuat Nasi Putih karena kamu tidak memiliki bahan berikut:
  1. Beras
=====
```

Gambar 6.8.4.5 Menjalankan fitur BOIL yang valid dan memilih salah satu makanan saat inventory tidak ada bahan yang dibutuhkan

Jika bahan makanan yang diperlukan ada, maka program akan menampilkan pesan bahwa makanan sudah berhasil direbus dan sudah dimasukkan dalam inventory serta langsung menghapus bahan makanan yang digunakan pada list inventory.

```
=====
=      BOIL      =
=====
List Bahan Makanan yang Bisa Dibuat:
  1. Nasi Putih
  2. Telur Rebus
  3. Air Panas

Kirim 0 untuk exit.
Enter Command: 1
Nasi Putih selesai dibuat dan sudah masuk ke inventory!
=====
E for EZ di posisi: (9,1)
Waktu saat ini: 00:03:29
Notifikasi:
1. Nasi Putih sudah diterima oleh BNMO
*****
*                                     *
*           T   X   X   X   *
*           X   X   X   X   *
*           X   X   X   *
*       M                                     *
*   X                                     *
*   X                                     *
*   X   X   F   C   *
*       S B           X   X   X   X   *
*           X                                     *
*           X   X   X   *
*           X                                     *
*           X   X   X   X   *
*       K                                     *
*           X   X   X   X   *
*           X   X   X   X   *
*****
```

Gambar 6.8.4.6 Menjalankan fitur BOIL yang valid dan inventory terdapat bahan makanan yang diperlukan untuk di-boil

6.9 Command WAIT

Pada Command Wait, program akan menerima input “WAIT X Y” dengan X adalah waktu dalam jam yang ingin ditambah dan Y adalah waktu dalam menit yang ingin ditambah. Jika input valid, program akan menampilkan waktu terbaru yaitu waktu sebelumnya ditambah dengan waktu yang ditambah dari command WAIT ini. Jika waktu yang diinput dalam command WAIT ini terlalu lama dan melebihi batas waktu kadaluarsa makanan, program akan menampilkan notifikasi bahwa makanan yang dibeli atau dibuat telah kadaluarsa.

```
=====
E for EZ di posisi: (1,5)
Waktu saat ini: 00:00:12
=====
Masukkan command: WAIT 8 8
=====
E for EZ di posisi: (1,5)
Waktu saat ini: 00:08:20
```

Gambar 6.9.1 Menjalankan fitur WAIT dengan menambah waktu sebanyak 8 jam dan 8 menit hingga toko buka

```
=====
E for EZ di posisi: (1,5)
Waktu saat ini: 01:08:43
=====
Masukkan command: WAIT 1000 20
=====
E for EZ di posisi: (1,5)
Waktu saat ini: 43:01:03
Notifikasi:
1. Beras telah kadaluarsa :(
2. Tepung telah kadaluarsa :(
3. Ayam Mentah telah kadaluarsa :(
4. Kunyit telah kadaluarsa :(
5. Santan telah kadaluarsa :(
6. Minyak Goreng telah kadaluarsa :(
7. Tempe Mentah telah kadaluarsa :(
8. Tahu Mentah telah kadaluarsa :(
9. Cabai telah kadaluarsa :(
```

Gambar 6.9.2 Menjalankan fitur WAIT dengan menambah waktu sebanyak 1000 jam dan 20 menit hingga makanan kadaluarsa

6.10 Command-Command Lain

6.10.1 Undo

Pada Command UNDO, program akan menerima input “UNDO”. Setelah input dimasukkan, program akan membatalkan command yang telah terjadi dan kembali pada kondisi satu langkah sebelum command dieksekusi.

6.10.1.1 Undo Buy dan Delivery

Proses BUY berhasil di-undo sehingga di delivery terdapat bahan makanan sebelumnya dan tidak terdapat makanan yang terakhir dibeli.

```
Masukkan ID makanan yang ingin dibeli: 3
Makanan berhasil dicheckout!
Silahkan menunggu makanan sampai di lokasi Anda!
=====
EZ di posisi: (1,5)
Waktu saat ini: 00:00:11
Notifikasi: -
=====
*
*      S T   X X X *
*      X X X X *
*      X X X X *
*      M      *
*      X      *
*      X      *
*      X X X   C   *
*      X X X   F   *
*      B      X X X X *
*      X      X      *
*      X X X   *
*      K      X X X X *
*      X X X X *
=====
Masukkan command:

Masukkan command: UNDO
Proses berhasil di-UNDO
=====
ez di posisi: (1,5)
Waktu saat ini: 00:00:10
=====
*
*      S T   X X X *
*      X X X X *
*      X X X X *
*      M      *
*      X      *
*      X      *
*      X X X   F   *
*      B      X X X X *
*      X      X      *
*      X X X   *
*      K      X X X X *
*      X X X X *
=====
Masukkan command: DELIVERY
Daftar barang yang sedang diantar:
1. Kunyit - 4 menit

=====
Masukkan command: DELIVERY
Daftar barang yang sedang diantar:
Tidak ada barang yang sedang diantar
```

Gambar 6.10.1.1.1 Menjalankan command UNDO setelah melakukan fitur BUY yang berhasil pada DELIVERY

6.10.1.2 Mix dan Inventory

Proses MIX berhasil di-undo sehingga inventory terdapat bahan makanan sebelumnya dan tidak terdapat makanan hasil MIX

```
Masukkan command: UNDO
Proses berhasil di-UNDO
=====
E for EZ di posisi: (3,3)
Waktu saat ini: 00:00:55
* * * * *
*           T   X   X   X   *
*           X   X   X   X   *
*       S       X   X   X   *
*       M       *
*   X           *
*   X           *
*   X   X   F   C   *
*       B       X   X   X   *
*           X       *
*           X   X   X   *
*       K       X       *
*           X   X   X   *
* * * * *
```

Gambar 6.10.1.2.1 Menjalankan fitur UNDO setelah berhasil mendapatkan makanan hasil proses MIX ke inventory

Inventory tidak terdapat makanan hasil mix dan bahan makanan sebelumnya kembali ke inventory.

```
Masukkan command: INVENTORY
Daftar isi INVENTORY:
1. Nasi Kuning - 1 hari
2. Kuyit - 17 hari 14 jam 21 menit
=====
E for EZ di posisi: (3,3)
Waktu saat ini: 00:00:55
Notifikasi: -
* * * * *
*           T   X   X   X   *
*           X   X   X   X   *
*       S       X   X   X   *
*       M       *
*   X           *
*   X           *
*   X   X   F   C   *
*       B       X   X   X   *
*           X       *
*           X   X   X   *
*       K       X       *
*           X   X   X   *
* * * * *
```

Gambar 6.10.1.2.2 Mengecek inventory setelah menjalankan fitur UNDO

6.10.1.3 Boil dan Inventory

Proses BOIL berhasil di-undo sehingga inventory terdapat bahan makanan sebelumnya dan tidak terdapat makanan hasil BOIL

```
Masukkan command: UNDO
Proses berhasil di-UNDO
=====
E for EZ di posisi: (9,1)
Waktu saat ini: 00:03:28
*****
*                               *
*           T   X   X   X   *
*         X X X   X   *
*         X   X   X   *
*       M                               *
*     X                               *
*     X                               *
*   X X X   F           C           *
*                               *
*   S B           X X X X   *
*                               *
*                               X   *
*                               X X X *
*                               X   *
*         K           X X X X   *
*                               *
*****
```

Gambar 6.10.1.3.1 Menjalankan fitur UNDO setelah berhasil mendapatkan makanan hasil proses BOIL

Inventory tidak terdapat makanan hasil boil dan bahan makanan sebelumnya kembali ke inventory

```
Masukkan command: INVENTORY
Daftar isi INVENTORY:
1. Beras - 29 hari 22 jam 59 menit
=====
E for EZ di posisi: (1,5)
Waktu saat ini: 00:03:18
Notifikasi: -
*****
*                               *
*           S T   X   X   X   *
*         X X X   X   *
*         X   X   X   *
*       M                               *
*     X                               *
*     X                               *
*   X X X   F           C           *
*                               *
*   B           X X X X   *
*                               *
*                               X   *
*                               X X X *
*                               X   *
*         K           X X X X   *
*                               *
*****
```

Gambar 6.10.1.3.1 Mengecek inventory setelah menjalankan fitur UNDO

6.10.1.4 Fry dan Inventory

Proses FRY berhasil di-undo sehingga inventory terdapat bahan makanan sebelumnya dan tidak terdapat makanan hasil FRY

```
Masukkan command: UNDO
Proses berhasil di-UNDO
=====
E for EZ di posisi: (9,1)
Waktu saat ini: 00:03:28
* * * * *
*           T   X   X   X   *
*           X   X   X   X   *
*           X   X   X   *
*       M           *
*   X           *
*   X           C   *
*   X X X   F           *
*       S B           X X X X *
*           X           *
*           X X X   *
*       K           X           *
*           X X X X *
* * * * *
```

Gambar 6.10.1.4.1 Menjalankan fitur UNDO setelah berhasil mendapatkan makanan hasil proses FRY

Inventory tidak terdapat makanan hasil fry dan bahan makanan sebelumnya kembali ke inventory

```
Masukkan command: INVENTORY
Daftar isi INVENTORY:
1. Nasi Goreng - 18 jam
2. Minyak Goreng - 9 hari 22 jam 22 menit
=====
E for EZ di posisi: (7,5)
Waktu saat ini: 00:03:45
Notifikasi: -
* * * * *
*           T   X   X   X   *
*           X   X   X   X   *
*           X   X   X   *
*       M           *
*   X           *
*   X           *
*   X X X   S F           *
*       B           X X X X *
*           X           *
*           X X X   *
*       K           X           *
*           X X X X *
* * * * *
```

Gambar 6.10.1.4.2 Mengecek inventory setelah menjalankan fitur UNDO

6.10.1.5 Chop dan Inventory

Proses CHOP berhasil di-undo sehingga inventory terdapat bahan makanan sebelumnya dan tidak terdapat makanan hasil CHOP

```
Masukkan command: UNDO
Proses berhasil di-UNDO
=====
E for EZ di posisi: (9,1)
Waktu saat ini: 00:03:28
*****
*                               *
*      T      X  X  X  *
*      X X X  X  *
*      X  X  X  *
*      M      *
*      X      *
*      X      *
*      X X X  F      C      *
*      S B      X X X X  *
*      *      X      *
*      K      X X X  *
*      X      *
*      X X X X  *
*      *      *
*****
```

Gambar 6.10.1.5.1 Menjalankan fitur UNDO setelah berhasil mendapatkan makanan hasil proses CHOP

Inventory tidak terdapat makanan hasil chop dan bahan makanan sebelumnya kembali ke inventory

```
Masukkan command: INVENTORY
Daftar isi INVENTORY:
1. Ayam Mentah - 21 hari 23 jam 9 menit
=====
E for EZ di posisi: (1,5)
Waktu saat ini: 00:01:44
Notifikasi: -
*****
*                               *
*      S T      X  X  X  *
*      X X X  X  *
*      X  X  X  *
*      M      *
*      X      *
*      X      *
*      X X X  F      C      *
*      B      X X X X  *
*      *      X      *
*      K      X X X  *
*      X      *
*      X X X X  *
*      *      *
*****
```

Gambar 6.10.1.5.2 Mengecek inventory setelah menjalankan fitur UNDO

6.10.1.6 Peta

Menjalankan command UNDO untuk mengembalikan simulator ke posisi sebelumnya

```
E for EZ di posisi: (0,0)
Waktu saat ini: 00:00:04
Notifikasi: -
*****
* 5 *****
*      T   X X X *
*      X X X X *
*      X X X *
*      H      *
*      X      *
*      X      *
*     X X X   C   *
*     F      *
*     B      X X X X *
*           X      *
*           X X X *
*          K      X *
*           X X X X *
*           *
*****

Masukkan command: UNDO
Proses berhasil di-UNDO

=====
E for EZ di posisi: (1,0)
Waktu saat ini: 00:00:05
*****
* 5 *****
*      T   X X X *
*      X X X X *
*      X X X *
*      H      *
*      X      *
*      X      *
*     X X X   C   *
*     F      *
*     B      X X X X *
*           X      *
*           X X X *
*          K      X *
*           X X X X *
*           *
*****

Masukkan command: 
```

Gambar 6.10.1.6.1 Proses simulator berhasil di UNDO

6.10.2 Redo

Pada Command REDO, program akan menerima input “REDO”. Setelah input dimasukkan, program akan membatalkan command UNDO yang sebelumnya dieksekusi dan kembali pada kondisi satu langkah setelah command UNDO.

6.10.2.1 Buy dan Delivery

Proses BUY berhasil di-redo sehingga di DELIVERY terdapat makanan yang terakhir dibeli.

```

=====
Masukkan command: REDO
Proses berhasil di-REDO
=====
EZ di posisi: (1,5)
Waktu saat ini: 00:00:11
Notifikasi: -
* * * * *
*           S T       X X X
*           X X X X
*           X X X
*           M
* X
* X           C
* X X X F
*
* B           X X X X
*           X
*           X X X
*           K       X
*           X X X X
*
* * * * *
=====
Masukkan command: DELIVERY
Daftar barang yang sedang diantar:
1. Kuyit - 4 menit

```

Gambar 6.10.2.1.1 Proses BUY berhasil di-redo sehingga di DELIVERY terdapat makanan yang terakhir dibeli

6.10.2.2 Mix dan Inventory

Proses MIX berhasil di-redo sehingga inventory terdapat makanan hasil MIX sebelumnya dan bahan makanan dihapus kembali dari inventory.

```
Masukkan command: REDO
Proses berhasil di-REDO
=====
E for EZ di posisi: (3,3)
Waktu saat ini: 00:00:56
Notifikasi: -
*****
*               *
*      T      X X X *
*      X X X X *
*      S      X X X *
*      M      *
*      X      *
*      X      *
*      X X X F      C *
*      B      X X X X *
*      K      X *
*      X X X *
*      X X X X *
*               *
*****
```

Gambar 6.10.2.2.1 Menjalankan fungsi REDO setelah berhasil dilakukan UNDO

Inventory kembali memiliki makanan hasil MIX dan bahan makanan sebelumnya dihapus dari inventory.

```
Masukkan command: INVENTORY
Daftar isi INVENTORY:
1. Nasi Kuning - 23 jam 59 menit
=====
E for EZ di posisi: (3,3)
Waktu saat ini: 00:00:56
Notifikasi: -
*****
*               *
*      T      X X X *
*      X X X X *
*      S      X X X *
*      M      *
*      X      *
*      X      *
*      X X X F      C *
*      B      X X X X *
*      K      X *
*      X X X *
*      X X X X *
*               *
*****
```

Gambar 6.10.2.2.2 Mengecek inventory setelah dilakukan REDO

6.10.2.3 Boil dan Inventory

Proses BOIL berhasil di-redo sehingga inventory terdapat makanan hasil BOIL sebelumnya dan bahan makanan dihapus kembali dari inventory.

```

Masukkan command: REDO
Proses berhasil di-REDO
=====
E for EZ di posisi: (3,3)
Waktu saat ini: 00:00:56
Notifikasi: -
*****
*                               *
*           T       X X X      *
*           X X X X      *
*           X X X      *
*       S           *
*       M           *
*   X           *
*   X           *
*   X X X       F       C       *
*       B           X X X X      *
*           X           *
*           X X X      *
*       K           X           *
*           X X X X      *
*                               *
*****

```

Gambar 6.10.2.3.1 Menjalankan fungsi REDO setelah berhasil dilakukan UNDO

Inventory kembali memiliki makanan hasil BOIL dan bahan makanan sebelumnya dihapus dari inventory.

```

Masukkan command: INVENTORY
Daftar isi INVENTORY:
1. Nasi Putih - 1 hari 11 jam 59 menit
=====
E for EZ di posisi: (9,1)
Waktu saat ini: 00:01:53
Notifikasi: -
*****
*                               *
*           T       X X X      *
*           X X X X      *
*           X X X      *
*       M           *
*   X           *
*   X           *
*   X X X       F       C       *
*       S B       X X X X      *
*           X           *
*           X X X      *
*       K           X           *
*           X X X X      *
*                               *
*****

```

Gambar 6.10.2.3.2 Mengecek inventory setelah dilakukan REDO

6.10.2.4 Fry dan Inventory

Proses FRY berhasil di-redo sehingga inventory terdapat makanan hasil FRY sebelumnya dan bahan makanan dihapus kembali dari inventory.

```

Masukkan command: REDO
Proses berhasil di-REDO
=====
E for EZ di posisi: (3,3)
Waktu saat ini: 00:00:56
Notifikasi: -
*****
*
*      T   X X X
*      XXXX
*      X X X
*
*      S
*      M
*
*      X
*      X
*      XXX   F      C
*
*      B      XXXXX
*      X
*      XXXX
*      X
*      K      XXXXX
*
*****

```

Gambar 6.10.2.4.1 Menjalankan fungsi REDO setelah berhasil dilakukan UNDO

Inventory kembali memiliki makanan hasil FRY dan bahan makanan sebelumnya dihapus dari inventory.

```

Masukkan command: INVENTORY
Daftar isi INVENTORY:
1. Nasi Goreng - 17 jam 59 menit
=====
E for EZ di posisi: (7,5)
Waktu saat ini: 00:03:46
Notifikasi: -
*****
*
*      T   X X X
*      XXXX
*      X X X
*
*      M
*
*      X
*      X
*      XXX   S F      C
*
*      B      XXXXX
*      X
*      XXXX
*      X
*      K      XXXXX
*
*****

```

Gambar 6.10.2.4.2 Mengecek inventory setelah dilakukan REDO

6.10.2.5 Chop dan Inventory

Proses CHOP berhasil di-redo sehingga inventory terdapat makanan hasil CHOP sebelumnya dan bahan makanan dihapus kembali dari inventory.

```

Masukkan command: REDO
Proses berhasil di-REDO
=====
E for EZ di posisi: (3,3)
Waktu saat ini: 00:00:56
Notifikasi: -
*****
*                               *
*           T   X   X   X   *
*           X   X   X   X   *
*           X   X   X   *
*       S           *
*       M           *
*   X           *
*   X           *
*   X X X   F           C   *
*       B           X X X X *
*           X           *
*           X X X   *
*       K           X           *
*           X X X X *
*                               *
*****

```

Gambar 6.10.2.5.1 Menjalankan fungsi REDO setelah berhasil dilakukan UNDO

Inventory kembali memiliki makanan hasil CHOP dan bahan makanan sebelumnya dihapus dari inventory.

```

Masukkan command: INVENTORY
Daftar isi INVENTORY:
1. Ayam Potong - 1 hari 5 jam
=====
E for EZ di posisi: (6,10)
Waktu saat ini: 00:01:54
Notifikasi: -
*****
*                               *
*           T   X   X   X   *
*           X   X   X   X   *
*           X   X   X   *
*       M           *
*   X           *
*   X           *
*   X X X   F           S C *
*       B           X X X X *
*           X           *
*           X X X   *
*       K           X           *
*           X X X X *
*                               *
*****

```

Gambar 6.10.2.5.2 Mengecek inventory setelah dilakukan REDO

6.10.2.6 Peta

Memasukkan command UNDO pada terminal. Prekonsisinya adalah harus ada proses yang bisa di redo (sebelumnya pernah di UNDO dan belum ada proses tambahan setelah di UNDO).

```
E for EZ di posisi: (1,0)
Waktu saat ini: 00:00:03
*****
*
* S      T   X X X *
*      XXX X *
*      X X X *
*
* M
* X
* X      C
* XXX F
*
* B      XXXX
*      X
*      XXX
*      K   X
*      XXXX
*
*****

Masukkan command: REDO
Proses berhasil di-REDO

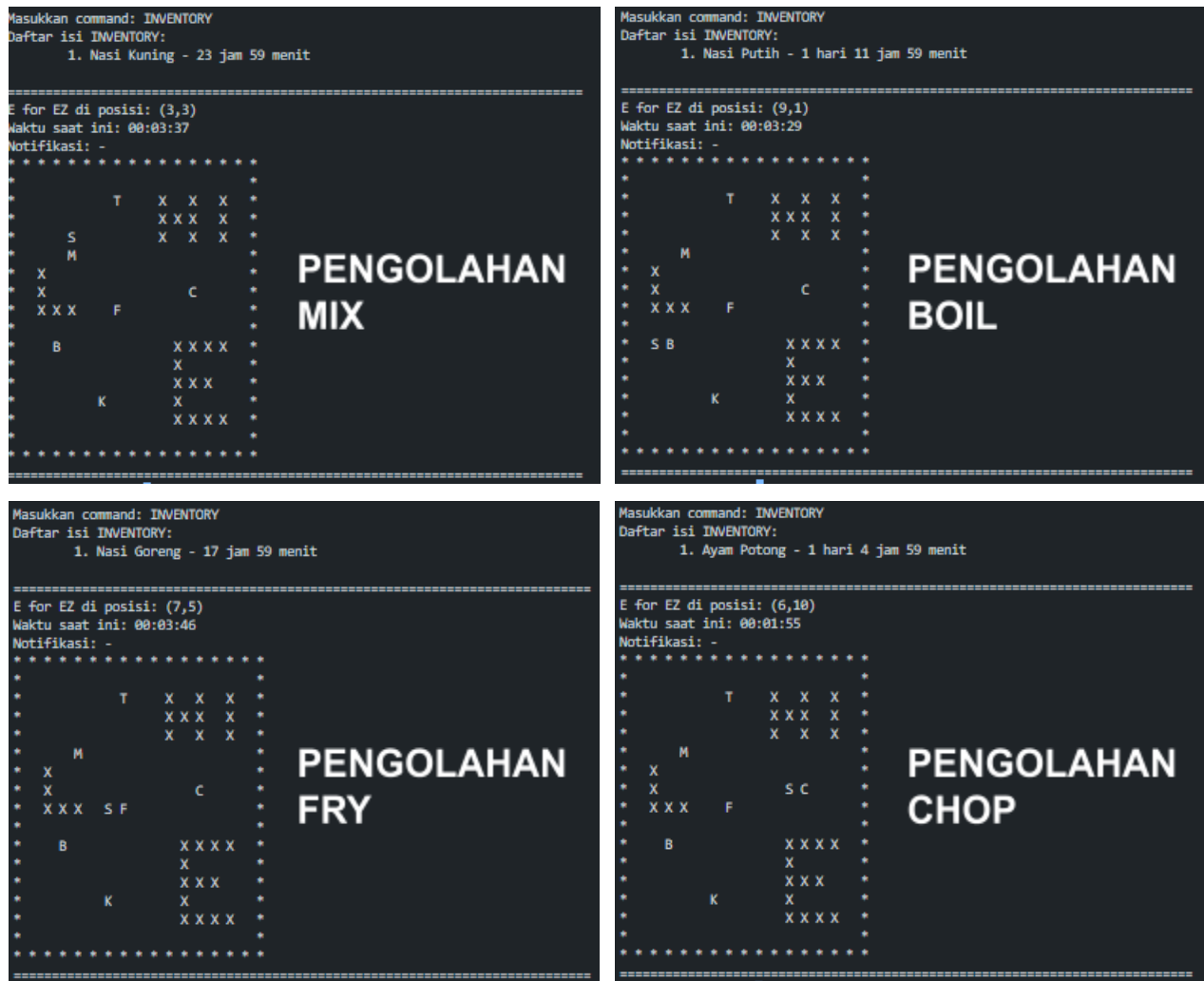
E for EZ di posisi: (0,0)
Waktu saat ini: 00:00:04
Notifikasi: -
*****
*
* S      T   X X X *
*      XXX X *
*      X X X *
*
* M
* X
* X      C
* XXX F
*
* B      XXXX
*      X
*      XXX
*      K   X
*      XXXX
*
*****

Masukkan command: 
```

Gambar 6.10.2.6.1 Proses simulator berhasil di REDO

6.10.3 Inventory

Pada Command INVENTORY, program akan menerima input “INVENTORY”. Setelah input dimasukkan, program akan langsung menampilkan semua bahan makanan yang sudah dibeli atau dibuat yang masuk kedalam inventory. Jika tidak ada makanan dalam inventory, program akan menampilkan bahwa inventory sedang kosong.



Gambar 6.10.3.1 Tampilan pengecekan inventory saat pengolahan makanan sudah berhasil

6.10.4 Catalog

Pada Command CATALOG, program akan menerima input “CATALOG”. Setelah input dimasukkan, program akan langsung menampilkan bahan dan makanan yang sedang tersedia dengan informasi detailnya seperti nomor ID makanan, waktu kadaluarsa, waktu pengiriman, aksi yang diperlukan untuk mendapatkan makanan, dan ukuran makanan.


```

=====
Masukkan command: CATALOG
01. Beras ----- 30:00:00 ----- Buy ----- 00:00:20 ----- (2,2)
02. Santan ----- 12:00:00 ----- Buy ----- 00:00:10 ----- (1,1)
03. Kunyit ----- 17:15:00 ----- Buy ----- 00:00:05 ----- (1,1)
04. Tepung ----- 25:10:00 ----- Buy ----- 00:00:10 ----- (1,2)
05. Minyak Goreng ----- 10:00:00 ----- Buy ----- 00:00:15 ----- (2,2)
06. Ayam Mentah ----- 22:00:00 ----- Buy ----- 00:00:30 ----- (2,2)
07. Cabai ----- 03:00:00 ----- Buy ----- 00:00:05 ----- (1,1)
08. Bawang ----- 00:12:00 ----- Buy ----- 00:00:05 ----- (1,1)
09. Tahu Mentah ----- 05:10:00 ----- Buy ----- 00:00:20 ----- (1,1)
10. Tempe Mentah ----- 05:10:00 ----- Buy ----- 00:00:20 ----- (1,1)
11. Telur ----- 05:10:00 ----- Buy ----- 00:00:05 ----- (1,1)
12. Bakso Mentah ----- 08:20:00 ----- Buy ----- 00:00:25 ----- (1,1)
13. Sosis Mentah ----- 08:20:00 ----- Buy ----- 00:00:25 ----- (1,2)
14. Air ----- 50:00:00 ----- Buy ----- 00:00:05 ----- (1,1)
15. Teh Sachet ----- 35:00:00 ----- Buy ----- 00:00:05 ----- (1,1)
16. Kopi Sachet ----- 35:00:00 ----- Buy ----- 00:00:05 ----- (1,1)
17. Es Batu Bongkahan ----- 00:00:30 ----- Buy ----- 00:00:05 ----- (1,1)
18. Nasi Putih ----- 01:12:00 ----- Boil ----- 00:00:35 ----- (1,1)
19. Nasi Kuning ----- 01:00:00 ----- Mix ----- 00:00:15 ----- (2,2)
20. Nasi Uduk ----- 01:00:39 ----- Mix ----- 00:01:00 ----- (2,2)
21. Nasi Goreng ----- 00:18:00 ----- Fry ----- 00:00:20 ----- (2,2)
22. Ayam Potong ----- 01:05:00 ----- Chop ----- 00:00:00 ----- (1,2)
23. Ayam Tepung ----- 00:15:00 ----- Mix ----- 00:00:00 ----- (3,3)
24. Ayam Goreng ----- 00:12:00 ----- Fry ----- 00:00:10 ----- (3,3)
25. Sambal ----- 01:12:00 ----- Mix ----- 00:00:00 ----- (2,2)
26. Sambal Goreng ----- 00:10:00 ----- Fry ----- 00:00:05 ----- (2,2)
27. Tahu Goreng ----- 00:20:00 ----- Fry ----- 00:00:08 ----- (2,2)
28. Tempe Goreng ----- 00:20:00 ----- Fry ----- 00:00:08 ----- (2,2)
29. Telur Dadar ----- 00:16:00 ----- Fry ----- 00:00:07 ----- (2,2)
30. Telur Rebus ----- 01:21:30 ----- Boil ----- 00:00:25 ----- (1,2)
31. Telur Balado ----- 00:15:30 ----- Mix ----- 00:00:05 ----- (2,3)
32. Bakso Goreng ----- 02:03:00 ----- Fry ----- 00:00:00 ----- (2,2)
33. Sosis Goreng ----- 02:03:00 ----- Fry ----- 00:00:00 ----- (2,2)
34. Air Panas ----- 00:00:30 ----- Boil ----- 00:00:00 ----- (1,1)
35. Teh Panas ----- 00:00:25 ----- Mix ----- 00:00:02 ----- (2,2)
36. Kopi Panas ----- 00:00:25 ----- Mix ----- 00:00:02 ----- (2,2)
37. Es Batu ----- 00:00:28 ----- Chop ----- 00:00:00 ----- (2,2)
38. Es Teh ----- 00:00:25 ----- Mix ----- 00:00:02 ----- (3,3)
39. Kopi Es ----- 00:00:25 ----- Mix ----- 00:00:02 ----- (3,3)
40. Ayam Goreng Extra Sambal ----- 00:11:30 ----- Mix ----- 00:00:15 ----- (4,4)
41. Nasi Putih dengan Ayam Goreng Extra Sambal ----- 00:11:00 ----- Mix ----- 00:00:20 ----- (4,4)
42. Nasi Kuning dengan Ayam Goreng Extra Sambal ----- 00:08:00 ----- Mix ----- 00:00:20 ----- (4,4)
43. Nasi Uduk dengan Ayam Goreng Extra Sambal ----- 00:08:00 ----- Mix ----- 00:00:20 ----- (4,4)
44. Nasi Goreng dengan Bakso ----- 00:15:30 ----- Mix ----- 00:00:20 ----- (4,4)
45. Nasi Goreng dengan Sosis ----- 00:15:30 ----- Mix ----- 00:00:20 ----- (4,4)
46. Nasi Goreng dengan Telur Dadar ----- 00:15:30 ----- Mix ----- 00:00:20 ----- (4,4)
47. Nasi Kuning dengan Telur Balado ----- 00:08:00 ----- Mix ----- 00:00:20 ----- (4,4)
=====
E for EZ di posisi: (12,4)
Waktu saat ini: 00:08:49
Notifikasi: -
* * * * *
*
*      T      X X X
*      X X X X
*      X X X
*      M
* X
* X      C
* X X X F
*
*      B      X X X X
*      X
*      X X X
*      S K      X
*      X X X X
*
* * * * *

```

Gambar 6.10.4.1 Tampilan saat menerima input “CATALOG”

6.10.5 COOKBOOK

Pada Command COOKBOOK, program akan menerima input “COOKBOOK”. Setelah input dimasukkan, program akan langsung menampilkan resep dari makanan yang tersedia.

```
=====
Masukkan command: COOKBOOK
01. Nasi Putih
   Boil ----- Beras
02. Nasi Kuning
   Mix ----- Nasi Putih ----- Kunyit
03. Nasi Uduk
   Mix ----- Nasi Putih ----- Santan
04. Nasi Goreng
   Fry ----- Nasi Putih ----- Minyak Goreng
05. Ayam Potong
   Chop ----- Ayam Mentah
06. Ayam Tepung
   Mix ----- Ayam Potong ----- Tepung
07. Ayam Goreng
   Fry ----- Ayam Tepung ----- Minyak Goreng
08. Sambal
   Mix ----- Cabai ----- Bawang
09. Sambal Goreng
   Fry ----- Sambal ----- Minyak Goreng
10. Tahu Goreng
   Fry ----- Tahu Mentah ----- Minyak Goreng
11. Tempe Goreng
   Fry ----- Tempe Mentah ----- Minyak Goreng
12. Telur Dadar
   Fry ----- Telur ----- Minyak Goreng
13. Telur Rebus
   Boil ----- Telur
14. Telur Balado
   Mix ----- Telur Rebus ----- Sambal Goreng
15. Bakso Goreng
   Fry ----- Bakso Mentah ----- Minyak Goreng
16. Sosis Goreng
   Fry ----- Sosis Mentah ----- Minyak Goreng
17. Air Panas
   Boil ----- Air
18. Teh Panas
   Mix ----- Air Panas ----- Teh Sachet
19. Kopi Panas
   Mix ----- Air Panas ----- Kopi Sachet
20. Es Batu
   Chop ----- Es Batu Bongkahan
21. Es Teh
   Mix ----- Es Batu ----- Teh Panas
22. Kopi Es
   Mix ----- Es Batu ----- Kopi Panas
23. Ayam Goreng Extra Sambal
   Mix ----- Sambal Goreng ----- Ayam Goreng
24. Nasi Putih dengan Ayam Goreng Extra Sambal
   Mix ----- Ayam Goreng Extra Sambal ----- Nasi Putih
25. Nasi Kuning dengan Ayam Goreng Extra Sambal
   Mix ----- Ayam Goreng Extra Sambal ----- Nasi Kuning
26. Nasi Uduk dengan Ayam Goreng Extra Sambal
   Mix ----- Ayam Goreng Extra Sambal ----- Nasi Uduk
27. Nasi Goreng dengan Bakso
   Mix ----- Bakso Goreng ----- Nasi Goreng
28. Nasi Goreng dengan Sosis
   Mix ----- Sosis Goreng ----- Nasi Goreng
29. Nasi Goreng dengan Telur Dadar
   Mix ----- Telur Dadar ----- Nasi Goreng
30. Nasi Kuning dengan Telur Balado
   Mix ----- Telur Balado ----- Nasi Kuning
```

Gambar 6.10.5.1 Tampilan saat menerima input “COOKBOOK”

6.10.6 Rekomendasi

Pada Command REKOMENDASI, program akan menerima input “REKOMENDASI”. Setelah input dimasukkan, program akan menampilkan daftar makanan yang direkomendasikan. Makanan yang direkomendasikan adalah makanan yang dapat diolah simulator pada saat inventory memiliki bahan makanan tertentu. Ketika inventory tidak memiliki bahan makanan apapun, program menyatakan bahwa tidak ada rekomendasi makanan. Sebaliknya, jika inventory terdapat setidaknya salah satu resep makanan yang lengkap, program menampilkan makanan yang dapat diolah simulator.

```
Masukkan command: REKOMENDASI
=== Berikut daftar makanan yang direkomendasikan untuk dibuat ===
Tidak ada makanan yang bisa direkomendasikan!
=====
E for EZ di posisi: (0,0)
Waktu saat ini: 00:00:00
Notifikasi: -
*****
* S                                     *
*           T   X   X   X   *
*           X   X   X   X   *
*           X   X   X   X   *
*           M                                     *
*           X                                     *
*           X                                     *
*           X   X   X   F   C   *
*           B           X   X   X   X   *
*           K           X   *
*           X   X   X   *
*           X   X   X   X   *
*****
```

Gambar 6.10.6.1 Mengecek command REKOMENDASI saat inventory kosong

```
Masukkan command: INVENTORY
Daftar isi INVENTORY:
1. Ayam Mentah - 21 hari 23 jam 9 menit
=====
E for EZ di posisi: (1,5)
Waktu saat ini: 00:01:42
Notifikasi: -
*****
*           S T   X   X   X   *
*           X   X   X   X   *
*           X   X   X   *
*           M                                     *
*           X                                     *
*           X                                     *
*           X   X   X   F   C   *
*           B           X   X   X   X   *
*           K           X   *
*           X   X   X   *
*           X   X   X   X   *
*****
```

```
Masukkan command: REKOMENDASI
=== Berikut daftar makanan yang direkomendasikan untuk dibuat ===
1. Ayam Potong
=====
E for EZ di posisi: (1,5)
Waktu saat ini: 00:01:42
Notifikasi: -
*****
*           S T   X   X   X   *
*           X   X   X   X   *
*           X   X   X   X   *
*           M                                     *
*           X                                     *
*           X                                     *
*           X   X   X   F   C   *
*           B           X   X   X   X   *
*           K           X   *
*           X   X   X   *
*           X   X   X   X   *
*****
```

Gambar 6.10.6.2 Mengecek command REKOMENDASI saat inventory memiliki setidaknya satu resep makanan

6.10.7 Kulkas

Command kulkas merupakan command yang akan dijalankan setelah pengguna menuliskan “KULKAS” pada program. Kulkas sendiri memiliki persyaratan tertentu sebelum dijalankan. Persyaratan ini adalah simulator harus berada di dekat kulkas. Setelah menjalankan fungsi kulkas, pengguna akan diminta untuk menginput 1 atau 2 dengan 1 merupakan pilihan untuk memasukkan makanan pada inventory ke dalam kulkas dan 2 merupakan pilihan untuk mengambil makanan dari kulkas.

Setelah memilih, program akan meminta pengguna untuk melakukan input makanan yang mana yang mau dimasukkan ke dalam kulkas atau diambil dari kulkas. Apabila makanan ingin dimasukkan ke dalam kulkas expired dari makanan akan selalu sama di dalam kulkas (tidak berubah) sedangkan jika pengguna ingin mengambil makanan dari kulkas maka program akan meminta pengguna untuk memasukkan input 1 untuk menaruhnya di inventory dan 2 untuk membuangnya.

```
=====
Masukkan command: KULKAS
=====
=      KULKAS      =
=====
1. Masukkan makanan ke dalam kulkas
2. Keluarkan makanan ke dalam kulkas

-----
Enter Command: 1
=====
Pilih makanan pada inventory :
Daftar isi INVENTORY:
  1. Bawang - 11 jam 37 menit
  2. Cabai - 2 hari 23 jam 36 menit
  3. Tahu Mentah - 5 hari 9 jam 53 menit
  4. Tempe Mentah - 5 hari 9 jam 54 menit
  5. Minyak Goreng - 9 hari 23 jam 44 menit
  6. Santan - 11 hari 23 jam 36 menit
  7. Kunyit - 17 hari 14 jam 32 menit
  8. Ayam Mentah - 22 hari
  9. Tepung - 25 hari 9 jam 38 menit
 10. Beras - 29 hari 23 jam 45 menit

-----

-----
Enter Command: 1
=====
[Empty box for food selection]
-----

Nama makanan = Bawang
Ukuran = (1,1)
Masukkan posisi koordinat yang akan ditempatkan untuk makanan!
NOTE : Untuk posisi bagian kiri atas makanan
-----
Enter Baris: 1
Enter Kolom: 1
=====
```

```

Enter Baris: 1
Enter Kolom: 1
=====
Berhasil Memasukkan Makanan Ke Dalam Kulkas!
=====
E for EZ di posisi: (12,4)
Waktu saat ini: 00:08:46
Notifikasi: -
* * * * *
*           T       X X X
*           X X X X
*           X X X
*           M
*      X
*      X           C
*    X X X   F
*
*      B           X X X X
*                   X
*                   X X X
*           S K     X
*                   X X X X
*
* * * * *
=====
Masukkan command:

```

Gambar 6.10.7.1 Memasukkan makanan ke dalam kulkas

```

=====
Masukkan command: KULKAS
=====
=      KULKAS      =
=====
1. Masukkan makanan ke dalam kulkas
2. Keluarkan makanan ke dalam kulkas

-----
Enter Command: 1
=====
Pilih makanan pada inventory :
Daftar isi INVENTORY:
1. Cabai - 2 hari 23 jam 35 menit
2. Tahu Mentah - 5 hari 9 jam 52 menit
3. Tempe Mentah - 5 hari 9 jam 53 menit
4. Minyak Goreng - 9 hari 23 jam 43 menit
5. Santan - 11 hari 23 jam 35 menit
6. Kunyit - 17 hari 14 jam 31 menit
7. Ayam Mentah - 21 hari 23 jam 59 menit
8. Tepung - 25 hari 9 jam 37 menit
9. Beras - 29 hari 23 jam 44 menit

-----
Enter Command: 7
=====

Enter Command: 7
=====
X

-----
Nama makanan = Ayam Mentah
Ukuran = (2,2)
Masukkan posisi koordinat yang akan ditempatkan untuk makanan!
NOTE : Untuk posisi bagian kiri atas makanan
-----
Enter Baris: 2
Enter Kolom: 2
=====

Enter Baris: 2
Enter Kolom: 2
=====
Berhasil Memasukkan Makanan Ke Dalam Kulkas!
=====
E for EZ di posisi: (12,4)
Waktu saat ini: 00:08:47
Notifikasi: -
* * * * *
*           T   X X X *
*           X X X X *
*           X X X *
*      M           *
* X             *
* X             C   *
* X X X   F           *
*           *
*      B           X X X X *
*           X           *
*           X X X *
*      S K           X *
*           X X X X *
*           *
* * * * *
=====
Masukkan command:

```

Gambar 6.10.7.2 Memasukkan makanan ke dalam kulkas yang sudah pernah diisi


```

=====
Masukkan command: KULKAS
=====
=      KULKAS      =
=====
1. Masukkan makanan ke dalam kulkas
2. Keluarkan makanan ke dalam kulkas

-----

Enter Command: 2
=====
Pilih makanan pada kulkas :

1.
ID makanan:      8
Nama makanan:     Bawang
Expired:          00:11:37
Delivery time:    00:00:05
Panjang (per satuan): 1
Lebar (per satuan): 1
Lokasi aksi:      Buy

2.
ID makanan:      6
Nama makanan:     Ayam Mentah
Expired:          21:23:59
Delivery time:    00:00:30
Panjang (per satuan): 2
Lebar (per satuan): 2
Lokasi aksi:      Buy
-----
Enter Command: 1
=====

Enter Command: 1
=====
Pilih opsi!

1. Simpan Makanan ke dalam Inventory
2. Buang Makanan
-----

Enter Command: 1
=====
Berhasil memindahkan isi kulkas ke dalam inventory!
=====

E for EZ di posisi: (12,4)
Waktu saat ini: 00:08:48
Notifikasi:
1. Bawang sudah diterima oleh BNMO
* * * * *
*           T       X X X *
*           X X X X *
*           X X X *
*           M       *
* X           *
* X           C       *
* X X X F       *
*           *
* B           X X X X *
*           X       *
*           X X X *
*           S K     X *
*           X X X X *
*           *
* * * * *
=====
Masukkan command:

```

Gambar 6.10.7.4 Mengambil makanan dari kulkas lalu menyimpannya ke dalam inventory


```

=====
Masukkan command: KULKAS
=====
=      KULKAS      =
=====
1. Masukkan makanan ke dalam kulkas
2. Keluarkan makanan ke dalam kulkas

-----

Enter Command: 2
=====

Pilih makanan pada kulkas :

1.
ID makanan:          6
Nama makanan:        Ayam Mentah
Expired:              21:23:59
Delivery time:        00:00:30
Panjang (per satuan): 2
Lebar (per satuan):   2
Lokasi aksi:          Buy
-----

Enter Command: 1
=====

Pilih opsi!

1. Simpan Makanan ke dalam Inventory
2. Buang Makanan
-----

Enter Command: 2
=====

Berhasil membuang isi kulkas!
=====

E for EZ di posisi: (12,4)
Waktu saat ini: 00:08:49
Notifikasi: -

```

Gambar 6.10.7.5 Mengambil makanan dari kulkas lalu membuangnya

7 Test Script

7.1 Test Command Inisiasi (START dan EXIT)

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	START	Memulai program dan menampilkan splash screen	Melakukan run pada program main kemudian langsung menampilkan splash screen	START	Memulai program	Gambar 6.5.1.1
2	EXIT	Mengakhiri program yang dijalankan	Menuliskan EXIT pada terminal	EXIT	Mengakhiri program	Gambar 6.5.2.1

7.2 Test Command Pemesanan (BUY dan DELIVERY)

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	BUY	Mengecek saat input valid	Menjalankan fitur BUY pada saat simulator berada didekat T(toko)	BUY	Program mengeluarkan output yang menampilkan daftar makanan yang dapat dibeli	Gambar 6.6.1.1
2	BUY	Mengecek saat input tidak valid	Menjalankan fitur BUY pada saat simulator berada tidak berada didekat T(toko)	BUY	Program gagal mengeluarkan output yang menampilkan daftar makanan yang dapat dibeli	Gambar 6.6.1.2

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
3	DELIVERY	Untuk mengecek daftar makanan yang sedang diantar.	Menjalankan fitur DELIVERY pada terminal pada saat belum ada yang di beli	DELIVERY	Program mengeluarkan output yang menampilkan tidak ada makanan yg sedang diantar	Gambar 6.6.2.1
4	BUY	Inputan valid dan membeli sesuatu	Memasukan ID makanan yang ingin dibeli	ID Makanan	Program menampilkan pesan bahwa makanan berhasil di checkout.	Gambar 6.6.1.5
5	DELIVERY	Untuk mengecek daftar makanan yang sedang diantar.	Menjalankan fitur DELIVERY pada terminal setelah membeli makanan	DELIVERY	Program mengeluarkan output yang menampilkan daftar makanan yg sedang diantar	Gambar 6.6.2.2
6	COMMAND LAIN	Menjalankan command UNDO setelah melakukan fitur BUY yang berhasil pada DELIVERY	Menjalankan fitur UNDO	UNDO DELIVERY	Proses BUY berhasil di-undo sehingga di delivery terdapat bahan makanan sebelumnya dan tidak terdapat	Gambar 6.10.1.1.1

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					makanan yang terakhir dibeli.	
7	COMMAN D LAIN	Proses BUY berhasil di-redo sehingga di DELIVERY terdapat makanan yang terakhir dibeli	Menjalankan fitur REDO	REDO DELIVERY	Proses BUY berhasil di-redo sehingga di DELIVERY terdapat makanan yang terakhir dibeli	Gambar 6.10.2.1.1
8	BUY	Mengecek saat input tidak valid	Menjalankan fitur BUY pada saat simulator berada didekat T(toko) dengan command yang salah	BUY	Program menampilkan pesan bahwa masukan tidak valid.	Gambar 6.6.1.3
9	BUY	Inputan valid tetapi ID tidak ditemukan	Memasukan ID makanan yang ingin dibeli tetapi ID tidak terdaftar dalam daftar menu.	ID Makanan	Program menampilkan "ID tidak ditemukan"	Gambar 6.6.1.4

7.3 Test Command COOKBOOK

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	COOKBOOK	Untuk mengecek resep makanan	Menjalankan fitur COOKBOOK pada terminal	COOKBOOK	Program mengeluarkan output yang menampilkan resep makanan	Gambar 6.10.5.1

7.4 Test Command WAIT

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	WAIT	Untuk mempercepat waktu	Menjalankan fitur WAIT pada terminal	WAIT X Y	Program mengeluarkan output yang menampilkan waktu terbaru	Gambar 6.9.1
2	WAIT	Mempercepat waktu hingga makanan melewati batas kadaluarsa	Menjalankan fitur WAIT pada terminal	WAIT X Y	Program mengeluarkan output yang menampilkan waktu terbaru ditambah notifikasi bahwa makanan telah kadaluarsa	Gambar 6.9.2

7.5 Test Command CATALOG

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	CATALOG	Untuk menampilkan bahan dan makanan yang tersedia.	Menjalankan command CATALOG pada terminal	CATALOG	Program mengeluarkan output yang menampilkan bahan dan makanan yang tersedia	Gambar 6.10.4.1

7.6 Test Command Pengolahan (MIX, CHOP, FRY, BOIL)

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	PENGOLAHAN	Mengecek saat input tidak valid	Menjalankan fitur MIX saat simulator tidak berada di dekat MIXER	MIX	Program mengeluarkan output yang menyatakan bahwa simulator tidak berada di dekat MIXER	Gambar 6.8.1.1
2	PENGOLAHAN	Mengecek saat input valid	Menjalankan fitur MIX saat simulator berada di dekat MIXER	MIX	Program mengeluarkan list makanan yang dapat diperoleh dengan cara MIX	Gambar 6.8.1.2
3	PENGOLAHAN	Mengecek saat simulator tidak memiliki bahan yang diperlukan	Menjalankan fitur MIX yang valid dan memilih salah satu makanan saat inventory tidak ada bahan yang dibutuhkan	MIX	Program mengeluarkan list bahan makanan yang diperlukan simulator dan tidak berada di inventory	Gambar 6.8.1.5
4	PENGOLAHAN	Mengecek saat input tidak terdapat di list makanan yang dapat di-mix	Menjalankan fitur MIX yang valid dan memilih angka yang tidak terdapat di list makanan yang dapat di-mix	MIX	Program melakukan looping hingga mendapat input yang valid	Gambar 6.8.1.4

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
5.	PENGOLAHAN	Mengecek input 0 saat program menampilkan menu MIX	Menjalankan fitur MIX yang valid dan memilih angka 0	MIX	Program mengeluarkan simulator dari menu MIX	Gambar 6.8.1.3
6.	PENGOLAHAN	Mengecek input yang berada di menu MIX dan inventory memiliki bahan yang diperlukan	Simulator mengumpulkan bahan makanan yang diperlukan lalu menjalankan fitur MIX yang valid dan inventory terdapat bahan makanan yang diperlukan untuk di-mix	MIX	Program berhasil menambah makanan yang diolah ke inventory dan menghapus bahan makanan yang terdapat di inventory	Gambar 6.8.1.6
7.	COMMAND LAIN	Mengecek inventory apakah sudah terdapat makanan yang di-mix dan bahan makanan dihapus dari inventory	Menjalankan fitur INVENTORY setelah berhasil mendapatkan makanan yang diolah dari fitur MIX yang valid	INVENTORY	Inventory terdapat makanan yang berhasil diolah dan bahan makanan tidak ada di inventory	Gambar 6.10.3.1
8.	COMMAND LAIN	Mengecek command UNDO setelah melakukan	Menjalankan fitur UNDO setelah berhasil mendapatkan makanan hasil proses MIX ke inventory	UNDO MIX	Proses MIX berhasil di-undo sehingga inventory	Gambar 6.10.1.2.1

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		fitur MIX yang berhasil			terdapat bahan makanan sebelumnya dan tidak terdapat makanan hasil MIX	
9.	COMMAND LAIN	Mengecek inventory setelah menjalankan fitur UNDO	Menjalankan fungsi INVENTORY setelah berhasil dilakukan UNDO	INVENTORY MIX	Inventory tidak terdapat makanan hasil mix dan bahan makanan sebelumnya kembali ke inventory	Gambar 6.10.1.2.2
10.	COMMAND LAIN	Mengecek fitur REDO setelah melakukan UNDO	Menjalankan fungsi REDO setelah berhasil dilakukan UNDO	REDO MIX	Proses MIX berhasil di-redo sehingga inventory terdapat makanan hasil MIX sebelumnya dan bahan makanan dihapus kembali dari inventory	Gambar 6.10.2.2.1

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
11.	COMMAN D LAIN	Mengecek inventory setelah dilakukan REDO	Menjalankan fitur INVENTORY setelah berhasil dilakukan REDO	INVENTORY MIX	Inventory kembali memiliki makanan hasil MIX dan bahan makanan sebelumnya dihapus dari inventory	Gambar 6.10.2.2.2
12.	PENGOLAHAN	Mengecek saat input tidak valid	Menjalankan fitur BOIL saat simulator tidak berada di dekat BOILER	BOIL	Program mengeluarkan output yang menyatakan bahwa simulator tidak berada di dekat BOILER	Gambar 6.8.4.1
13.	PENGOLAHAN	Mengecek saat input valid	Menjalankan fitur BOIL saat simulator berada di dekat BOILER	BOIL	Program mengeluarkan list makanan yang dapat diperoleh dengan cara BOIL	Gambar 6.8.4.2
14.	PENGOLAHAN	Mengecek saat simulator tidak memiliki bahan yang diperlukan	Menjalankan fitur BOIL yang valid dan memilih salah satu makanan saat inventory tidak ada bahan yang dibutuhkan	BOIL	Program mengeluarkan list bahan makanan yang diperlukan simulator dan	Gambar 6.8.4.5

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					tidak berada di inventory	
15.	PENGOLAHAN	Mengecek saat input tidak terdapat di list makanan yang dapat di-boil	Menjalankan fitur BOIL yang valid dan memilih angka yang tidak terdapat di list makanan yang dapat di-boil	BOIL	Program melakukan looping hingga mendapat input yang valid	Gambar 6.8.4.4
16.	PENGOLAHAN	Mengecek input 0 saat program menampilkan menu BOIL	Menjalankan fitur BOIL yang valid dan memilih angka 0	BOIL	Program mengeluarkan simulator dari menu BOIL	Gambar 6.8.4.3
17.	PENGOLAHAN	Mengecek input yang berada di menu BOIL dan inventory memiliki bahan yang diperlukan	Simulator mengumpulkan bahan makanan yang diperlukan lalu menjalankan fitur BOIL yang valid dan inventory terdapat bahan makanan yang diperlukan untuk di-boil	BOIL	Program berhasil menambah makanan yang diolah ke inventory dan menghapus bahan makanan yang terdapat di inventory	Gambar 6.8.4.6
18.	COMMAND LAIN	Mengecek inventory apakah sudah terdapat makanan yang di-boil	Menjalankan fitur INVENTORY setelah berhasil mendapatkan makanan yang diolah dari fitur BOIL yang valid	INVENTORY BOIL	Inventory terdapat makanan yang berhasil diolah dan bahan makanan tidak	Gambar 6.10.3.1

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		dan bahan makanan dihapus dari inventory			ada di inventory	
19.	COMMAND LAIN	Mengecek command UNDO setelah melakukan fitur BOIL yang berhasil	Menjalankan fitur UNDO setelah berhasil mendapatkan makanan hasil proses BOIL	UNDO BOIL	Proses BOIL berhasil di-undo sehingga inventory terdapat bahan makanan sebelumnya dan tidak terdapat makanan hasil BOIL	Gambar 6.10.1.3.1
20.	COMMAND LAIN	Mengecek inventory setelah menjalankan fitur UNDO	Menjalankan fungsi INVENTORY setelah berhasil dilakukan UNDO	INVENTORY BOIL	Inventory tidak terdapat makanan hasil boil dan bahan makanan sebelumnya kembali ke inventory	Gambar 6.10.1.3.2
21.	COMMAND LAIN	Mengecek fitur REDO setelah melakukan UNDO	Menjalankan fungsi REDO setelah berhasil dilakukan UNDO	REDO BOIL	Proses BOIL berhasil di-redo sehingga inventory terdapat makanan hasil BOIL sebelumnya	Gambar 6.10.2.3.1

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					dan bahan makanan dihapus kembali dari inventory	
22.	COMMAN D LAIN	Mengecek inventory setelah dilakukan REDO	Menjalankan fitur INVENTORY setelah berhasil dilakukan REDO	INVENTORY BOIL	Inventory kembali memiliki makanan hasil BOIL dan bahan makanan sebelumnya dihapus dari inventory	Gambar 6.10.2.3.2
23.	PENGOLAHAN	Mengecek saat input tidak valid	Menjalankan fitur FRY saat simulator tidak berada di dekat FRYER	FRY	Program mengeluarkan output yang menyatakan bahwa simulator tidak berada di dekat FRYER	Gambar 6.8.3.1
24.	PENGOLAHAN	Mengecek saat input valid	Menjalankan fitur FRY saat simulator berada di dekat FRYER	FRY	Program mengeluarkan list makanan yang dapat diperoleh dengan cara FRY	Gambar 6.8.3.2
25.	PENGOLAHAN	Mengecek saat simulator	Menjalankan fitur FRY yang valid dan memilih salah satu makanan saat	FRY	Program mengeluarkan list bahan	Gambar 6.8.3.5

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		tidak memiliki bahan yang diperlukan	inventory tidak ada bahan yang dibutuhkan		makanan yang diperlukan simulator dan tidak berada di inventory	
26.	PENGOLAHAN	Mengecek saat input tidak terdapat di list makanan yang dapat di-fry	Menjalankan fitur FRY yang valid dan memilih angka yang tidak terdapat di list makanan yang dapat di-fry	FRY	Program melakukan looping hingga mendapat input yang valid	Gambar 6.8.3.4
27.	PENGOLAHAN	Mengecek input 0 saat program menampilkan menu FRY	Menjalankan fitur FRY yang valid dan memilih angka 0	FRY	Program mengeluarkan simulator dari menu FRY	Gambar 6.8.3.3
28.	PENGOLAHAN	Mengecek input yang berada di menu FRY dan inventory memiliki bahan yang diperlukan	Simulator mengumpulkan bahan makanan yang diperlukan lalu menjalankan fitur FRY yang valid dan inventory terdapat bahan makanan yang diperlukan untuk di-fry	FRY	Program berhasil menambah makanan yang diolah ke inventory dan menghapus bahan makanan yang terdapat di inventory	Gambar 6.8.3.6
29.	COMMAND LAIN	Mengecek inventory apakah sudah	Menjalankan fitur INVENTORY setelah berhasil mendapatkan	INVENTORY FRY	Inventory terdapat makanan yang berhasil diolah	Gambar 6.10.3.1

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		terdapat makanan yang di-fry dan bahan makanan dihapus dari inventory	makanan yang diolah dari fitur FRY yang valid		dan bahan makanan tidak ada di inventory	
30.	COMMAND LAIN	Mengecek command UNDO setelah melakukan fitur FRY yang berhasil	Menjalankan fitur UNDO setelah berhasil mendapatkan makanan hasil proses FRY	UNDO FRY	Proses FRY berhasil di-undo sehingga inventory terdapat bahan makanan sebelumnya dan tidak terdapat makanan hasil FRY	Gambar 6.10.1.4.1
31.	COMMAND LAIN	Mengecek inventory setelah menjalankan fitur UNDO	Menjalankan fungsi INVENTORY setelah berhasil dilakukan UNDO	INVENTORY FRY	Inventory tidak terdapat makanan hasil fry dan bahan makanan sebelumnya kembali ke inventory	Gambar 6.10.1.4.2
32.	COMMAND LAIN	Mengecek fitur REDO setelah melakukan UNDO	Menjalankan fungsi REDO setelah berhasil dilakukan UNDO	REDO FRY	Proses FRY berhasil di-redo sehingga inventory terdapat	Gambar 6.10.2.4.1.

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					makanan hasil FRY sebelumnya dan bahan makanan dihapus kembali dari inventory	
33.	COMMAN D LAIN	Mengecek inventory setelah dilakukan REDO	Menjalankan fitur INVENTORY setelah berhasil dilakukan REDO	INVENTORY FRY	Inventory kembali memiliki makanan hasil FRY dan bahan makanan sebelumnya dihapus dari inventory	Gambar 6.10.2.4.2
34.	PENGOLAHAN	Mengecek saat input tidak valid	Menjalankan fitur CHOP saat simulator tidak berada di dekat CHOPPER	CHOP	Program mengeluarkan output yang menyatakan bahwa simulator tidak berada di dekat CHOPPER	Gambar 6.8.2.1
35.	PENGOLAHAN	Mengecek saat input valid	Menjalankan fitur CHOP saat simulator berada di dekat CHOPPER	CHOP	Program mengeluarkan list makanan yang dapat diperoleh	Gambar 6.8.2.2

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					dengan cara CHOP	
36.	PENGOLAHAN	Mengecek saat simulator tidak memiliki bahan yang diperlukan	Menjalankan fitur CHOP yang valid dan memilih salah satu makanan saat inventory tidak ada bahan yang dibutuhkan	CHOP	Program mengeluarkan list bahan makanan yang diperlukan simulator dan tidak berada di inventory	Gambar 6.8.2.5
37.	PENGOLAHAN	Mengecek saat input tidak terdapat di list makanan yang dapat di-chop	Menjalankan fitur CHOP yang valid dan memilih angka yang tidak terdapat di list makanan yang dapat di-chop	CHOP	Program melakukan looping hingga mendapat input yang valid	Gambar 6.8.2.4
38.	PENGOLAHAN	Mengecek input 0 saat program menampilkan menu CHOP	Menjalankan fitur CHOP yang valid dan memilih angka 0	CHOP	Program mengeluarkan simulator dari menu CHOP	Gambar 6.8.2.3
39.	PENGOLAHAN	Mengecek input yang berada di menu CHOP dan inventory memiliki bahan yang diperlukan	Simulator mengumpulkan bahan makanan yang diperlukan lalu menjalankan fitur CHOP yang valid dan inventory terdapat bahan makanan yang diperlukan untuk di-chop	CHOP	Program berhasil menambah makanan yang diolah ke inventory dan menghapus bahan makanan yang	Gambar 6.8.2.6

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					terdapat di inventory	
40.	COMMAND LAIN	Mengecek inventory apakah sudah terdapat makanan yang di-chop dan bahan makanan dihapus dari inventory	Menjalankan fitur INVENTORY setelah berhasil mendapatkan makanan yang diolah dari fitur CHOP yang valid	INVENTORY CHOP	Inventory terdapat makanan yang berhasil diolah dan bahan makanan tidak ada di inventory	Gambar 6.10.3.1
41.	COMMAND LAIN	Mengecek command UNDO setelah melakukan fitur CHOP yang berhasil	Menjalankan fitur UNDO setelah berhasil mendapatkan makanan hasil proses CHOP	UNDO CHOP	Proses CHOP berhasil di-undo sehingga inventory terdapat bahan makanan sebelumnya dan tidak terdapat makanan hasil CHOP	Gambar 6.10.1.5.1
42.	COMMAND LAIN	Mengecek inventory setelah menjalankan fitur UNDO	Menjalankan fungsi INVENTORY setelah berhasil dilakukan UNDO	INVENTORY CHOP	Inventory tidak terdapat makanan hasil chop dan bahan makanan sebelumnya	Gambar 6.10.1.5.2

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					kembali ke inventory	
43.	COMMAN D LAIN	Mengecek fitur REDO setelah melakukan UNDO	Menjalankan fungsi REDO setelah berhasil dilakukan UNDO	REDO CHOP	Proses CHOP berhasil di-redo sehingga inventory terdapat makanan hasil CHOP sebelumnya dan bahan makanan dihapus kembali dari inventory	Gambar 6.10.2.5.1
44.	COMMAN D LAIN	Mengecek inventory setelah dilakukan REDO	Menjalankan fitur INVENTORY setelah berhasil dilakukan REDO	INVENTORY CHOP	Inventory kembali memiliki makanan hasil CHOP dan bahan makanan sebelumnya dihapus dari inventory	Gambar 6.10.2.5.2

7.7 Test Command Peta (MOVE NORTH, MOVE EAST, MOVE WEST, MOVE SOUTH)

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Lokasi	Mengecek command move yang tidak valid untuk bergerak ke utara	Memasukkan command MOVE NORTH pada terminal	MOVE NORTH	Simulator Gagal berpindah	Gambar 6.7.1.1
2	Lokasi	Mengecek command move yang valid untuk bergerak ke timur	Memasukkan command MOVE EAST pada terminal	MOVE EAST	Simulator Berhasil berpindah	Gambar 6.7.2.1
3	Lokasi	Mengecek command move yang valid untuk bergerak ke selatan	Memasukkan command MOVE SOUTH pada terminal	MOVE SOUTH	Berhasil berpindah	Gambar 6.7.4.1
4	Lokasi	Mengecek command move yang valid untuk bergerak ke barat	Memasukkan command MOVE WEST pada terminal	MOVE WEST	Simulator berhasil berpindah	Gambar 6.7.3.1
5	Lokasi	Mengecek command move yang valid untuk	Memasukkan command MOVE NORTH pada terminal	MOVE NORTH	Simulator berhasil berpindah	Gambar 6.7.1.2

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		bergerak ke utara				
6	Command Lain	Mengecek command UNDO untuk mengembalikan simulator ke posisi sebelumnya	Memasukkan command UNDO pada terminal	UNDO	Proses simulator berhasil di UNDO	Gambar 6.10.1.6.1
7	Command Lain	Mengecek command REDO untuk mengembalikan simulator ke posisi setelahnya	Memasukkan command UNDO pada terminal. Prekonsisinya adalah harus ada proses yang bisa di redo (sebelumnya pernah di UNDO dan belum ada proses tambahan setelah di UNDO).	REDO	Proses simulator berhasil di REDO	Gambar 6.10.2.6.1

7.8 Test Command Rekomendasi (BONUS)

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1.	BONUS	Mengecek command REKOMENDASI saat inventory kosong	Memanggil command REKOMENDASI saat program baru dijalankan	REKOMENDASI	Program mengeluarkan output yang menyatakan bahwa tidak ada makanan yang dapat direkomendasikan	Gambar 6.10.6.1

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
2,	BONUS	Mengecek command REKOMENDASI saat inventory memiliki setidaknya satu resep makanan	Memanggil command REKOMENDASI saat inventory memiliki satu bahan lengkap	REKOMENDASI	Program mengeluarkan output list rekomendasi makanan yang dapat diolah simulator	Gambar 6.10.6.2

7.9 Test Command Kulkas (BONUS)

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1.	BONUS	Menyimpan makanan ke dalam kulkas dengan input valid	Memiliki isi inventory lalu simulator berada di dekat kulkas dan memanggil fungsi KULKAS	KULKAS, pilihan 1, nomor makanan yang mau disimpan, koordinat valid.	Program mengeluarkan output yang menyatakan bahwa makanan berhasil dimasukkan ke dalam kulkas.	Gambar 6.10.7.1
2.	BONUS	Menyimpan makanan ke dalam kulkas dengan input valid dan kulkas	Memiliki isi inventory lalu simulator berada di dekat kulkas dan memanggil fungsi KULKAS	KULKAS, pilihan 1, nomor makanan yang mau disimpan, koordinat valid.	Program mengeluarkan output yang menyatakan bahwa makanan berhasil	Gambar 6.10.7.2

		sudah pernah diisi			dimasukkan ke dalam kulkas.	
3.	BONUS	Menyimpan makanan ke dalam kulkas dengan input tidak valid dan kulkas sudah pernah diisi	Memiliki isi inventory lalu simulator berada di dekat kulkas dan memanggil fungsi KULKAS	KULKAS, pilihan 1, nomor makanan yang mau disimpan, koordinat tidak valid.	Program mengeluarkan output yang menyatakan bahwa makanan gagal dimasukkan ke dalam kulkas.	Gambar 6.10.7.3
4.	BONUS	Mengambil makanan dari kulkas lalu ditaruh di inventory	Memiliki isi kulkas lalu simulator berada di dekat kulkas dan memanggil fungsi KULKAS	KULKAS, pilihan 2, nomor makanan yang mau diambil, pilihan 1.	Program mengeluarkan output yang menyatakan bahwa makanan berhasil dimasukkan ke inventory.	Gambar 6.10.7.4
5.	BONUS	Mengambil makanan dari kulkas lalu ditaruh di inventory	Memiliki isi kulkas lalu simulator berada di dekat kulkas dan memanggil fungsi KULKAS	KULKAS, pilihan 2, nomor makanan yang mau diambil, pilihan 2.	Program mengeluarkan output yang menyatakan bahwa makanan berhasil dibuang.	Gambar 6.10.7.5

8 Pembagian Kerja dalam Kelompok

NIM - Nama	Tugas
13521008 - Jason Rivalino	Readfile makanan, driver, dataset makanan, dataset resep, merancang tree untuk program, laporan, kulkas
13521015 - Hidayatullah Wildan Ghaly B	Simulator, Tree, Peta, Rekomendasi, Cookbook, Catalog, Point, Undo-Redo, Queue, Mesin Input, List Statik, Notifikasi, Driver, Kulkas.
13521022 - Raditya Naufal Abiyu	Splash screen intro, Splash screen exit, pemesanan, delivery, buy, merapikan struktur program, bug hunting, driver, laporan, kulkas
13521029 - M. Malik I. Baharsyah	ADT dan driver Makanan, pengolahan (mix, fry, chop, boil), laporan, kulkas
13521030 - Jauza Lathifah Annassalafi	ADT dan driver Time, Wait dan driver, bug tester, laporan, kulkas

9 Lampiran

9.1 Deskripsi Tugas Besar

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*”.

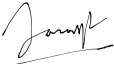





Namun, ada masalah. BNMO tidak tahu cara memasak dan Doni tidak bisa membantu persiapan karena ada hal lain. BNMO tidak bisa belajar dari video *youcub* karena BNMO adalah sebuah komputer sehingga hal yang paling mudah untuk dilakukan adalah membuat program simulasi untuk ditiru BNMO. Oleh karena itu,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

9.2 Notulen Rapat

9.2.1 Asistensi 1

Form Asistensi Tugas Besar
IF2110/Algoritma dan Struktur Data
Sem. 1 2021/2022

No. Kelompok/Kelas : E/K3
Nama Kelompok : E for EZ
Anggota Kelompok (Nama/NIM) :
1. Jason Rivalino / 13521008
2. Hidayatullah Wildan Ghaly B./ 13521015
3. Raditya Naufal Abiyu / 13521022
4. M. Malik I. Baharsyah / 13521029
5. Jauza Lathifah Annassalafi / 13521030
Asisten Pembimbing : Fabian Savero Diaz Pranoto
Asistensi I

Tanggal : 24 Oktober 2022	Catatan Asistensi:
Tempat : Zoom meeting	
Kehadiran Anggota Kelompok: <div style="text-align: center;">1 13521008 </div> <div style="text-align: center;">2 13521015 </div> <div style="text-align: center;">3 13521022 </div> <div style="text-align: center;">4 13521029 </div> <div style="text-align: center;">5 13521030 </div>	<ol style="list-style-type: none"> 1. Apa boleh menggunakan NULL atau scanf dalam pengerjaan tubes? <ul style="list-style-type: none"> - NULL dan scanf tidak diperlukan karena semua operasi pada string dilakukan pada mesin kata. 2. File konfigurasi itu isinya apa aja? <ul style="list-style-type: none"> - Isinya berupa daftar menu, bentuk peta, dan tree 3. File konfigurasi boleh dipisah tidak? <ul style="list-style-type: none"> - Untuk mempermudah boleh dipisah 4. Misalkan ada tempat yang tidak bisa diakses karena dibatasi tembok, apakah perlu ada handling untuk ini? <ul style="list-style-type: none"> - Tidak perlu, karena peta dibebaskan 5. Penilaian berdasarkan apa? <ul style="list-style-type: none"> - Lebih ke demo, untuk kodingannya nanti akan dicek apakah sesuai dengan spek dan batasan atau tidak 6. Penilaian bonus seberapa besar? <ul style="list-style-type: none"> - Bonus itu untuk nambahin nilai, nilai spek wajib itu jauh lebih besar jadi usahain selesain spek wajib terlebih dahulu. 7. Apakah input peta akan error apabila dibawah 10 x 10? <ul style="list-style-type: none"> - Peta minimalnya 10 x 10 jadi harus disesuaikan dengan spek
	Tanda Tangan Asisten: 

9.2.2 Notulen Rapat 1

RAPAT 1 K3 KELOMPOK E ALSTRUKDAT

Hari/tanggal	Jumat, 21 Oktober 2022
Waktu	15.00 - 15:30
Agenda	Membahas spek tubes alstrukdat
	Membahas cara pengerjaan, asistensi, dll
Anggota yang hadir	Willy
	Jason
	Radit
	Jauza
	Malik

Hasil Rapat:

1. Pengerjaan melalui github (**Radit** yang buat)
2. Isi meet-meet asistensi 1 di <https://www.when2meet.com/?17377940-Jn44n> usahain pilih secepatnya biar bisa langsung di request asistensi ke asisten
3. Tadi udah ngebahas spek tugas dll dan bisa kalian cek lagi di link tubes
4. Drive tubes ini udah berisi semua info tubes, asistensi, laporan, dan notulensi. Bisa kalian edit sendiri kalau misal ada yang mau ditambahin (contoh: nyicil laporan)
<https://drive.google.com/drive/folders/1oW6Nz6dW9sW8PIWd6SI9mV0XRim9HgGw?usp=sharing>
5. Pembahasan singkat tentang tubes:
 - Kita buat peta dengan matrix $n \times m$
 - Gerakan kita bisa dideteksi dengan command

- Ngelakuin sesuatu harus di tempatnya (misal masak harus di tempat masak)
 - Kalau item kurang berarti output “tidak berhasil dilakukan”
 - Kalau berhasil maka proses dilanjutkan, dan seterusnya
 - Barang yang kita miliki disimpan dalam inventory
6. Jangan lupa buat **driver** fungsi ADT
 7. Pembagian tugas akan dilakukan **setelah asistensi pertama**

9.2.3 Notulen Rapat 2

RAPAT 2 K3 KELOMPOK E ALSTRUKDAT

Hari/tanggal	Jumat, 28 Oktober 2022
Waktu	19.00 - 20.20
Agenda	Membahas progres pembagian tugas 1
	Membahas pembagian tugas tahap 2
Anggota yang hadir	Willy
	Jason
	Radit
	Malik

Hasil Rapat:

1. ADT MAKANAN (Malik) :

```
#include prio
```

```
{ array atau stack atau list of id }
```

```
void createMakanan (makanan *food, word nama, time exp, time deliver, word aksi)
```

```
void displayMakanan (makanan food)
```

void dealloc (prio *P)

void isValidFood (prio *P)

{print makanan basi kalau exp time ≤ 0 lalu deque}

void minusTime(prio *P, int selama_apa)

{semua makanan yang ada di queue di -1 expired timenya}







{lalu cek kalau masih > 0 biarin, kalau ≤ 0 di deque}

{dalam prio itu time utama + waktu expired}

2. ADT TREE (Jason), header basicnya dulu
3. ADT SIMULATOR (Willy), Point pisahin sama matrix
4. START (Radit), QUIT (Radit) dibuat splash screen
5. Laporan (Jauza + siapapun yang gabut)

9.2.4 Asistensi 2

Asistensi II

Tanggal : 1 November 2022	Catatan Asistensi: Dilakukan progres report kepada asisten. ADT Boleh banyak dan ADT boleh dipisah-pisah di file yang berbeda. ADT Tree boleh memanfaatkan konsep child and sibling selain pointer of pointer dan list dinamik.
Tempat : Zoom meeting	
Kehadiran Anggota Kelompok: 1 13521008  2 13521015  3 13521022  4 13521029  5 13521030 	
	Tanda Tangan Asisten: 

9.2.5 Notulen Rapat 3

RAPAT 3 K3 KELOMPOK E ALSTRUKDAT

Hari/tanggal	Sabtu, 5 November 2022
Waktu	13.00-14.00
Agenda	Membahas pembagian tugas tahap 3
Anggota yang hadir	Willy
	Jason
	Radit
	Jauza
	Malik

Hasil Rapat:

1. Membagi tugas berdasarkan Command yang belum selesai dan pengerjaan Bonus
2. Command Buy dan Delivery (Radit)
 - void BUY (ListStatik daftar, queue *Delivery)
 - void DELIVERY (queue Delivery)
 - I.S. Delivery sembarang
 - F.S. Nampilin isi dari Delivery
3. Command Pengolahan (Mix, Chop, Fry, Boil) (Malik)
 - Bisa pakai isClose(UserPeta(sim), '<char_tempat>')
4. Command Time (Jauza)
 - >> WAIT x y
 - void WAIT (TIME *RealTime, int x, int y)
5. Command Undo/Redo (Willy)
 - Buat ADT baru tipe stack yang berisi:

- RealTime
- Simulator
- Delivery
- Kulkas


6. Bonus Kulkas (Jason)

- ADT Matrix of makanan dengan size tertentu

9.3 Log Activity Anggota Kelompok

No	Tanggal	Tempat	NIM Anggota	Kegiatan
1	20 Oktober 2022	-	13521008 13521015 13521022 13521029 13521030	Pembuatan grup dan penentuan meet pertama serta pembuatan drive terpusat untuk tubes kali ini
2	21 Oktober 2022	Zoom meeting	13521008 13521015 13521022 13521029 13521030	Pembahasan spek tubes alstrukdat. Notulensi: Notulensi Rapat 1
3	24 Oktober 2022	Zoom meeting	13521008 13521015 13521022 13521029 13521030	Asistensi 1 Memastikan kebenaran spek tubes alstukdat dengan kakak asisten serta menanyakan beragam pertanyaan mengenai tubes ini.

4	24 Oktober 2022	Zoom meeting	13521008 13521015 13521022 13521029 13521030	Pembagian tugas tahap pertama untuk masing-masing anggota kelompok
5	25 Oktober 2022	-	13521015	Menyelesaikan ADT matrix dan fungsi primitifnya. Menyelesaikan ADT mesin kata dan implementasinya pada ADT string. Menyelesaikan bagian pembacaan peta dan MOVE serta command parser . Membuat dan menyelesaikan ADT stack khusus integer untuk membantu pengerjaan.
6	26 Oktober 2022	Kos	13521030	Menyelesaikan ADT time dan fungsi primitifnya untuk dapat dimanfaatkan di ADT lainnya.
7	27 Oktober 2022	Zoom meeting	13521015 13521008	Membahas cara pembuatan pembacaan file untuk makanan
8	28 Oktober 2022	Zoom meeting	13521008 13521015 13521022	Rapat kedua untuk membahas progress yang sudah dilakukan serta pembagian tugas tahap kedua.

			13521029	Notulensi:  Notulensi Rapat 2
9	28 Oktober 2022	-	13521022	Menyelesaikan splash screen yang sangat keren dan command EXIT .
10	28 Oktober 2022	-	13521008	Membuat fungsi pembacaan makanan dari file.
11	29 Oktober 2022		13521015	Membuat dan menyelesaikan ADT priority queue dan fungsi primitifnya untuk menyimpan makanan yang telah dibaca dari file untuk command CATALOG
12	29 Oktober 2022	-	13521015	Membuat dan menyelesaikan ADT simulator dan fungsi primitifnya.
13	30 Oktober 2022	-	13521029	Membuat dan menyelesaikan ADT makanan dan fungsi primitifnya.
14	30 Oktober 2022	Zoom meeting	13521015 13521008	Membahas dan mencari ide untuk pembuatan ADT tree dan fungsi primitifnya.
15	30 Oktober 2022	-	13521015	Membuat dan menyelesaikan ADT stack notifikasi serta primitifnya untuk menyimpan setiap notifikasi yang ada setiap pemanggilan command. Membuat dan menyelesaikan ADT point dan primitifnya untuk mengetahui posisi simulator.

16	31 Oktober 2022	Offline	13521015 13521008	Membandingkan dan membahas kode ADT Tree yang telah dibuat.
17	31 Oktober 2022	Kos dan Djoeroe coffee	13521015 13521008 13521030	Membuat laporan.
18	31 Oktober 2022	Kos	13521008	Membuat dan menyelesaikan file makanan dan file resep makanan.
19	1 November 2022	-	13521015	Membuat dan menyelesaikan ADT Tree dan fungsi primitifnya.
20	2 November 2022	-	13521015	Membuat dan menyelesaikan driver dari ADT tree serta pembuatan pembacaan resep makanan dari file.
21	2 November 2022	-	13521008	Membuat dan menyelesaikan gambar pohon resep makanan untuk.
22	3 November 2022	-	13521015	Melakukan fixing program yang terjadi segmentation fault dalam ADT Tree dan menyelesaikan fungsi merge tree untuk menggabungkan dua atau lebih tree. Dengan memanfaatkan fungsi ini, resep pada file akan membentuk pohon yang sempurna.
23	3 November 2022	-	13521022	Membuat pemanggilan fungsi splash screen pada main.
24	4 November 2022	-	13521015	Membuat dan menyelesaikan ADT

				LIST STATIK dan fungsi primitifnya serta fungsi sorting berdasarkan ID untuk menyempurnakan catalog.
25	5 November 2022	Zoom meeting	13521008 13521015 13521022 13521029 13521030	Rapat ketiga untuk membahas pembagian tugas tahap 3 dan pembahasan tentang pembuatan bonus.
26	5 November 2022	-	13521015	Membuat dan menyelesaikan fungsi COOKBOOK dan menyempurnakan ADT Simulator dan queueFood.
27	6 November 2022	-	13521015	Membuat dan menyelesaikan ADT stack of proses dan fungsi primitifnya serta fungsi UNDO dan REDO .
28	7 November 2022	-	13521015	Membuat ADT stackRekomendasi untuk menyimpan daftar makanan yang direkomendasikan.
29	8 November 2022	-	13521015	Membuat dan menyelesaikan ADT REKOMENDASI dan fungsi primitifnya serta membuat implementasinya dengan driver.
30	9 November 2022	-	13521022	Membuat dan menyelesaikan command BUY .
31	9 November 2022	-	13521030	Membuat dan menyelesaikan fungsi WAIT dan drivernya.

32	10 November 2022	-	13521029	Membuat dan menyelesaikan fungsi PENGOLAHAN MAKANAN .
33	12 November 2022	-	13521015	Fixing error pada semua file header.
34	13 November 2022	-	13521030	Membuat driver untuk command WAIT .
35	13 November 2022	-	13521015	Membuat dan menyelesaikan ADT sentencemachine untuk input pada MAIN dan fungsi primitifnya serta drivernya.
36	13 November 2022	-	13521022	Membuat pemanggilan fungsi splash pada MAIN .
37	14 November 2022	Zoom meeting	13521008 13521015 13521022 13521029 13521030	Membuat MAIN .
38	15 November 2022	-	13521029	Revisi PENGOLAHAN MAKANAN .
39	16 November 2022	-	13521015	Menyempurnakan MAIN dan saat ini program sudah bisa dijalankan .
40	18 November 2022	Kantin GKU 2	13521008 13521015 13521022 13521029	Melanjutkan pembuatan laporan.

			13521030	
41	19 November	Koojai Coffee	13521008 13521015 13521022 13521029 13521030	Melanjutkan pembuatan laporan dan driver.
42	20 November	Zoom Meeting	13521008 13521015 13521022 13521029 13521030	Menyelesaikan pembuatan laporan, KULKAS, dan uji coba program.

9.4 Milestone 1

Tanggal		DD-MM-YYYY
No	Fitur	Progress (0-100%)
1.	Command Parser	100% Pada 25 Oktober 2022
2.	Inisiasi	67% Pada 28 Oktober 2022
	a. Splash Screen	100% Pada 28 Oktober 2022
	b. Command START	langsung implementasi di main
	c. Command EXIT	100% Pada 28 Oktober 2022
3.	Simulator	100% Pada 29 Oktober 2022

	a. ADT Simulator	100% Pada 29 Oktober 2022
4.	Makanan	100% Pada 30 Oktober 2022
	a. Membaca makanan dari file	100% Pada 28 Oktober 2022
	b. ADT Makanan	100% Pada 30 Oktober 2022
	c. Command CATALOG	100% Pada 28 Oktober 2022
6.	Peta	100% Pada 25 Oktober 2022
	a. Membaca peta dari file	100% Pada 25 Oktober 2022
	b. Command MOVE NORTH/EAST/SOUTH/WEST	100% Pada 25 Oktober 2022
7.	Mekanisme Waktu	50% Pada 26 Oktober 2022
	a. ADT Waktu	100% Pada 26 Oktober 2022
	b. Waktu bertambah seiring command yg valid	langsung implementasi di main
8.	Laporan (50%)	100% Pada 31 Oktober 2022