

# **LAPORAN TUGAS BESAR**

## **IF2110/Algoritma dan Struktur Data**


### **Mesin BNMO**

Dipersiapkan oleh:

Kelompok A

Rizky Abdilah R	13521109
Shelma Salsabila	13521115
Juan Christopher S	13521116
Ahmad Ghulam	13521118
Raynard Tanadi	13521143
Muhammad Zaki A	13521146

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung  
Jl. Ganesha 10, Bandung 40132

	<b>Sekolah Teknik Elektro dan Informatika ITB</b>	<b>Nomor Dokumen</b>		<b>Halaman</b>
		<i>IF2110-TB-A-02</i>		<i>71</i>
		<i>Revisi</i>	<i>0</i>	<i>20 November 2022</i>

## Daftar Isi

1.	Ringkasan.....	4
1.1	Deskripsi Umum Persoalan.....	4
1.2	Isi Umum Laporan .....	4
1.3	Kesimpulan Hasil Tugas Besar .....	5
2	Penjelasan Tambahan Spesifikasi Tugas .....	5
2.1	Bonus 1: Kulkas .....	5
2.1.2	Mekanisme Penyimpanan.....	5
2.1.3	Mekanisme Tata Letak .....	6
2.2	Bonus 2: Pengolahan Makanan.....	6
2.2.1	Penjelasan Singkat.....	6
2.2.2	Mekanisme Penggunaan .....	6
3	Struktur Data (ADT) .....	6
3.1	Point .....	6
3.2	Time .....	8
3.3	Makanan .....	9
3.4	Stack .....	11
3.5	Mesin Karakter dan Mesin Kata.....	11
3.6	String .....	13
3.7	Simulator .....	14
3.8	Resep .....	15
3.9	ListStatik .....	15
3.10	Tree .....	17
3.11	Prioqueue .....	18
3.12	Matrix .....	
4	Program Utama .....	20
5	Algoritma-Algoritma Menarik.....	21
5.1	Kulkas.....	21
5.2	Mekanisme Penyesuaian Waktu pada PriorityQueue .....	21
6	Data Test .....	22
6.1	Start, Konfigurasi, dan Main Menu.....	22
6.2	Move.....	23
6.2.1	Move North.....	24
6.2.2	Move South.....	25
6.2.3	Move West .....	26
6.2.4	Move East .....	27
6.2.5	Invalid Move atau Invalid Input .....	28
6.3	Buy .....	29
6.4	Mix .....	31
6.5	Fry .....	33
6.6	Chop.....	35
6.7	Boil.....	37
6.8	Catalog .....	38
6.9	Cookbook .....	39
6.10	Inventory.....	39

6.11	Delivery .....	40
6.12	Process .....	40
6.13	Wait (X)(Y) .....	41
6.14	Undo Redo.....	45
6.15	Fridge.....	48
6.15.1	Fridge Show .....	49
6.15.2	Fridge Take .....	49
6.15.3	Fridge Put.....	50
6.15.4	Invalid Input.....	51
7	Test Script .....	52
8	Pembagian Kerja dalam Kelompok .....	58
9	Lampiran .....	60
9.1	Deskripsi Tugas Besar 2.....	60
9.2	Notulen Rapat.....	60
9.3	Log Activity Anggota Kelompok.....	71

# 1. Ringkasan

## 1.1 Deskripsi Umum Persoalan

Mesin BNMO adalah suatu mesin berbasis CLI (*command-line interface*) yang dibuat dengan menggunakan bahasa C dan memanfaatkan Struktur Data (*Abstract Data Type*) untuk memudahkan pemrograman. Mesin BNMO merupakan mesin dimana mesin itu dapat menjalankan suatu proses yang berhubungan dengan pengolahan makanan. Tugas mesin BNMO ini secara umum adalah mengolah suatu bahan menjadi makanan. Namun, tidak hanya itu masih ada fungsi lainnya.

Mesin BNMO ini dapat dimanfaatkan untuk membuat suatu makanan. Mesin ini diprogram untuk bisa memasak namun tidak hanya itu, membeli suatu bahan, mixing, merebus, memotong, menampilkan makanan yang ada serta menunjukkan kadaluarsa suatu makanan. Mesin dapat bergerak dalam suatu peta yang di dalam peta itu terdapat daerah dimana mesin dapat melakukan suatu aksi terhadap makanan.

Adapun ADT yang harus digunakan dalam pembuatan mesin ini adalah sebagai berikut.

- ADT sederhana termasuk di dalamnya ADT point, waktu, makanan dan simulator.
- ADT list statik
- ADT matriks
- ADT mesin karakter dan mesin kata
- ADT queue dengan pendekatan array list dinamik
- ADT stack
- ADT tree
- Serta ADT tambahan jika memang dibutuhkan

## 1.2 Isi Umum Laporan

Secara umum, laporan Tugas Besar IF2110 Kelompok A kelas K2 terdiri dari 9 bagian

1. Bagian 1: Berisi deskripsi umum dan penjelasan tentang mesin BNMO, pemaparan isi laporan secara umum, serta kesimpulan mengenai Tugas Besar IF2110.
2. Bagian 2: Berisi penjelasan tambahan yang diperlukan terkait spesifikasi tugas.
3. Bagian 3: Berisi penjelasan dari setiap Struktur Data yang digunakan serta kegunaannya sesuai dengan spesifikasi yang ada.
4. Bagian 4: Berisi penjelasan mengenai algoritma program utama dimulai dari START hingga EXIT.
5. Bagian 5: Berisi penjelasan mengenai algoritma yang dianggap menarik oleh kelompok kami beserta dengan penggunaannya dalam program.
6. Bagian 6: Berisi penjelasan mengenai fitur-fitur yang akan dicoba dan hasil yang diharapkan dari program.
7. Bagian 7: Berisi Tabel *Test Script* yang terdiri dari fitur yang di tes, tujuan testing, langkah-langkah testing, input data test, hasil yang diharapkan serta hasil yang keluar.
8. Bagian 8: Tabel pembagian kerja kelompok A kelas K2

STEI- ITB	IF2110_TB_02_A	Halaman 4 dari 71 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.		

9. Bagian 9: Lampiran-lampiran seperti notulensi dan lainnya.

### 1.3 **Kesimpulan Hasil Tugas Besar**

Tugas besar IF2110 cukup menarik untuk dikerjakan, mengeksplor hal baru serta membuat ADT dan implementasi fungsi baru menjadi hal yang menarik. Beberapa ADT yang harus dimodifikasi serta pengaplikasian ADT yang semua harus dimanfaatkan menjadi suatu tantangan tersendiri. Efisiensi dari setiap ADT serta keefektifannya menjadi suatu hal yang harus dipertimbangkan ketika suatu ADT itu akan digunakan. Untuk hasil tugas besar yang dihasilkan cukup baik, semua program bekerja dan berjalan semestinya.

## 2 **Penjelasan Tambahan Spesifikasi Tugas**

### 2.1 **Bonus 1: Kulkas**

#### 2.1.1 **Penjelasan Singkat**

ADT Kulkas dibuat dengan berdasar pada ADT Matrix, ADT Makanan, dan ADT Point. ADT Matrix digunakan sebagai fondasi untuk bentuk kulkas. Kulkas yang dibuat pada program adalah matrix berukuran  $\text{maxRow} \times \text{maxCol}$  (dengan setelan default  $\text{maxRow} = 5$  dan  $\text{maxCol} = 10$ ) yang tiap elemennya berisi tipe bentukan *IsiKulkas*. *IsiKulkas* adalah tipe bentukan yang terdiri dari Makanan (tipe Makanan dari ADT Makanan) dan Identifier (tipe integer). Aspek makanan pada *IsiKulkas* tentunya adalah makanan yang ingin disimpan pada kulkas tersebut. Di sisi lain, aspek identifier pada *IsiKulkas* merupakan integer yang digunakan untuk membedakan suatu makanan dengan makanan lainnya pada kulkas tersebut. Terakhir, ADT Point digunakan sebagai penunjuk koordinat pada kulkas yang berbentuk matrix. ADT Point inilah yang digunakan user untuk menunjuk dipetak manakah makanan akan diletakkan atau diambil.

#### 2.1.2 **Mekanisme Penyimpanan**

Terdapat 3 *commands* yang dapat digunakan *user* dalam menggunakan ADT Kulkas, yaitu:

- FRIDGE SHOW
- FRIDGE TAKE
- FRIDGE PUT

Semua command yang berkaitan dengan Kulkas dianggap tidak menghabiskan waktu. *Command* FRIDGE SHOW digunakan hanya untuk melihat *IsiKulkas*, tidak untuk mengambil ataupun menaruh makanan. *Command* FRIDGE TAKE digunakan untuk mengambil makanan pada kulkas. Dalam hal ini program hanya bisa dilakukan apabila kulkas tidak kosong. Terakhir, *Command* FRIDGE PUT digunakan untuk menaruh makanan pada kulkas. Seperti halnya FRIDGE TAKE, *command* ini juga hanya bisa dilakukan apabila *inventory* tidak kosong.

### 2.1.3 Mekanisme Tata Letak

Dalam menyimpan atau mengambil makanan pada Kulkas, program akan meminta koordinat petak pada kulkas kepada *user* tempat sebuah makanan disimpan atau akan disimpan. Program lantas akan melakukan pengecekan apakah koordinat yang dimasukkan bersifat valid atau tidak.

Ketika meletakkan makanan, bila makanan yang akan dimasukkan memiliki *size* lebih dari 1x1, maka petak yang dipilih oleh *user* akan menjadi petak paling kiri atas makanan pada kulkas. Program juga akan mengecek apakah dengan petak yang diinput *user* dan ukuran makanan, makanan tersebut valid untuk diletakkan (tidak keluar batasan petak kulkas).

Ketika mengambil makanan, jika petak yang diinput *user* bersifat tidak valid atau merupakan petak kosong, maka tidak ada makanan yang diambil dari kulkas. Bila koordinat yang diberikan bersifat valid dan ada makanan pada petak tersebut, tetapi *size* makanan lebih dari 1x1, maka program akan memasukkan makanan pada *inventory*, lalu menghapus semua data makanan pada kulkas yang memiliki *identifier* yang sama dengan makanan yang diambil.

## 2.2 Bonus 2: Pengolahan Makanan

### 2.2.1 Penjelasan Singkat

Pengolahan makanan akan memakan waktu berbeda-beda sesuai dengan data pada tipe bentukan Makanan masing-masing. Untuk makanan yang didapat melalui *command buy*, makanan akan masuk ke *delivery list* terlebih dahulu selama waktu yang telah ditetapkan pada aspek *delivery*. Namun, untuk makanan-makanan yang tidak bisa didapat melalui *command buy*, aspek *delivery* akan diisi dengan lamanya proses yang dibutuhkan untuk membuat makanan tersebut. Lantas, untuk makanan yang tidak didapat melalui *command buy*, makanan akan masuk ke *process list* terlebih dahulu selama rentang waktu tertentu sebelum masuk ke dalam *inventory*.

### 2.2.2 Mekanisme Penggunaan

Pada saat suatu makanan akan dibuat, program akan melakukan cek terlebih dahulu apakah pada *inventory* simulator terdapat bahan-bahan yang diperlukan. Bila simulator tidak memiliki bahan yang dibutuhkan, maka diberikan pesan bahwa simulator tidak memiliki bahan. Di sisi lain, bila kondisi tersebut terpenuhi, maka bahan-bahan pada *inventory* simulator akan diambil, lalu makanan yang akan dibuat dimasukkan pada *process list*.

## 3 Struktur Data (ADT)

### 3.1 Point

ADT point terdiri dari dua nilai yaitu x sebagai Absis dan y sebagai Ordinat, ADT ini digunakan untuk menunjukkan koordinat suatu mesin berada. Koordinat ini bisa berubah sesuai dengan posisi mesin BNMO. Alasan menggunakan ADT ini adalah mudahnya untuk

STEI- ITB	IF2110_TB_02_A	Halaman 6 dari 71 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.		

menentukan lokasi berdasarkan nilai matriks yang memiliki nilai kolom dan baris yang sesuai dengan Absis dan Ordinat. ADT ini diimplementasikan sebagai POINT di file point.c. Program ini memiliki beberapa primitif, yaitu:

1. Selektor :
  - Absis(P)
  - Ordinat(P)
2. Konstruktor :
  - void CreatePoint (POINT \* P, int X, int Y)  
Membentuk sebuah POINT dari komponen-komponennya
3. Fungsi/Procedure lain :
  - void BacaPOINT (POINT \* P)  
Membaca nilai absis dan ordinat dari keyboard dan membentuk POINT P berdasarkan dari nilai absis dan ordinat tersebut
  - void TulisPOINT (POINT P);  
Nilai P ditulis ke layar dengan format "(X,Y)"
  - boolean EQ (POINT P1, POINT P2)  
Mengirimkan true jika P1 = P2 : absis dan ordinatnya sama
  - boolean NEQ (POINT P1, POINT P2)  
Mengirimkan true jika P1 tidak sama dengan P2
  - boolean IsOrigin (POINT P)  
Menghasilkan true jika P adalah titik origin
  - boolean IsOnSbX (POINT P)  
Menghasilkan true jika P terletak Pada sumbu X
  - boolean IsOnSbY (POINT P)  
Menghasilkan true jika P terletak pada sumbu Y
  - int Kuadran (POINT P)  
Menghasilkan kuadran dari P: 1, 2, 3, atau 4  
Prekondisi : P bukan titik origin, dan P tidak terletak di salah satu sumbu
  - POINT NextX (POINT P)  
Mengirim salinan P dengan absis ditambah satu
  - POINT NextY (POINT P)  
Mengirim salinan P dengan ordinat ditambah satu
  - POINT BackX (POINT P)  
Mengirim salinan P dengan absis ditambah satu
  - POINT BackY (POINT P)  
Mengirim salinan P dengan ordinat ditambah satu
  - POINT PlusDelta (POINT P, int deltaX, int deltaY)  
Mengirim salinan P yang absisnya adalah Absis(P) + deltaX dan ordinatnya adalah Ordinat(P) + deltaY
  - POINT MirrorOf (POINT P, boolean SbX)  
Menghasilkan salinan P yang dicerminkan terhadap salah satu sumbu Jika SbX bernilai true, maka dicerminkan terhadap sumbu X Jika SbX bernilai false, maka dicerminkan terhadap sumbu Y

- void Geser (POINT \*P, int deltaX, int deltaY)  
I.S. P terdefinisi  
F.S. P digeser, absisnya sebesar deltaX dan ordinatnya sebesar deltaY
- void GeserKeSbX (POINT \*P)  
I.S. P terdefinisi  
F.S. P berada pada sumbu X dengan absis sama dengan absis semula.  
Proses : P digeser ke sumbu X.
- void GeserKeSbY (POINT \*P)  
I.S. P terdefinisi  
F.S. P berada pada sumbu Y dengan ordinat yang sama dengan semula.  
Proses : P digeser ke sumbu Y.
- void Mirror (POINT \*P, boolean SbX)  
I.S. P terdefinisi  
F.S. P dicerminkan tergantung nilai SbX atau SbY Jika SbX true maka dicerminkan terhadap sumbu X Jika SbX false maka dicerminkan terhadap sumbu Y

### 3.2 Time

ADT time terdiri dari tiga nilai yaitu Day, Hour, dan Minute. ADT ini digunakan untuk menunjukan waktu baik itu waktu delivery, waktu memasak, menggoreng, waktu kadaluarsa suatu makanan dan lainnya. Alasan menggunakan ADT ini adalah karena mudah digunakan serta untuk membuatnya cukup mudah karena telah dipelajari di kelas. ADT ini diimplementasikan dalam TIME di file time.c. Program ini memiliki beberapa primitif yaitu :

#### 4. Selektor

- Day(T)
- Hour(T)
- Minute(T)

#### 5. Konstruktor

- a. void createTime (TIME \* T, int DD, int HH, int MM)  
Membentuk sebuah TIME dari komponen-komponennya yang valid.  
Prekondisi : DD, HH, MM valid untuk membentuk TIME.

#### 6. Fungsi/Procedure lain

- b. boolean IsTIMEValid (int H, int M, int D);  
Mengirim true jika DD,HH,MM dapat membentuk TIME yang valid \*/ dipakai untuk mentest SEBELUM membentuk sebuah type time
- c. void BacaTIME (TIME \* T);  
Proses pembacaan time yang terdiri atas DD,HH dan MM Kalo gak valid di ulangi sampai valid yang terdiri dari DD,HH,MM digunain buat nunjukin kadaluarsa dari suatu makanan.
- d. void TulisTIMEDelivery (TIME T)  
I.S. T sembarang  
Proses : menulis nilai setiap komponen T ke layar dalam format ("HH.MM") tanpa karakter apa pun di depan atau belakangnya, termasuk spasi, enter, dll.
- e. void TulisTIME(TIME T)



- Menulis TIME dalam format <DD,HH,MM>
- f. void TulisTIMEString(TIME T);  
Menulis TIME dalam format DD hari HH jam MM menit
- g. TIME NextMinute (TIME T);  
Mengirim 1 menit setelah T dalam bentuk TIME ini berfungsi untuk proses delivery pada proses ini hari tidak ikut ditampilkan.
- h. TIME NextNMinute (TIME T, int N)  
Mengirim N menit setelah T dalam bentuk TIME
- i. TIME PrevMinute (TIME T)  
Mengirim 1 menit sebelum T dalam bentuk TIME
- j. TIME PrevNMinute (TIME T, int N)  
Mengirim N menit sebelum T dalam bentuk TIME
- k. boolean TEQ (TIME T1, TIME T2)  
Mengirimkan true jika T1=T2, false jika tidak
- l. boolean TNEQ (TIME T1, TIME T2)  
Mengirimkan true jika T1 tidak sama dengan T2
- m. boolean TLT (TIME T1, TIME T2)  
Mengirimkan true jika T1<T2, false jika tidak
- n. boolean TGT (TIME T1, TIME T2);  
Mengirimkan true jika T1>T2, false jika tidak
- o. TIME roundToEvenHours(TIME T);  
Membulatkan TIME ke jam genap terdekat
- p. int TIMEtoint (TIME T);  
Mengubah dari TIME ke bentuk int menit
- q. TIME intoTIME (int n);  
Mengubah dari int menit ke bentuk TIME

### 3.3 Makanan

ADT makanan terdiri dari:

- ID makanan (*id*),
- nama makanan (*name*),
- waktu kadaluarsa (*expired*),
- aksi yang akan digunakan untuk mendapatkan makanan (*action*),
- ukuran makanan (*x size* dan *y size* makanan untuk ADT Kulkas) (*area*), dan
- waktu *delivery* atau proses suatu makanan (*delivery*).

ADT ini bisa dikatakan merupakan ADT utama karena tipe makanan hampir digunakan pada setiap fitur yang ada. Alasan memilih ADT ini makanan yang memiliki info yang banyak menjadi mudah direpresentasikan. ADT ini diimplementasikan dalam Makanan di file makanan.c.

Untuk aspek ukuran makanan (*area*), *data type* Makanan dibantu oleh suatu *data type* lain yang dinamakan Size. Size terdiri dari dua aspek, yaitu *xSize* dan *ySize*, yang masing-masing merepresentasikan panjang dan lebar makanan saat ditaruh pada kulkas.

- Selektor:
  - idMkn(M)

- Selektor ini digunakan untuk menunjuk aspek ID pada *data type* Makanan. *Command* ini ekuivalen dengan (M).id.
- nameMkn(M)  
Selektor ini digunakan untuk menunjuk aspek nama pada *data type* Makanan. *Command* ini ekuivalen dengan (M).name.
  - expMkn(M)  
Selektor ini digunakan untuk menunjuk aspek waktu kadaluarsa pada *data type* Makanan. *Command* ini ekuivalen dengan (M).expired.
  - actMkn(M)  
Selektor ini digunakan untuk menunjuk aspek aksi makanan pada *data type* Makanan. *Command* ini ekuivalen dengan (M).action.
  - sizeMkn(M)  
Selektor ini digunakan untuk menunjuk aspek ukuran pada *data type* Makanan. *Command* ini ekuivalen dengan (M).area.
  - xSizeMkn(M)  
Selektor ini digunakan untuk menunjuk aspek panjang makanan pada *data type* Makanan saat diletakkan pada kulkas. *Command* ini ekuivalen dengan (M).area.xSize.
  - ySizeMkn(M)  
Selektor ini digunakan untuk menunjuk aspek lebar makanan pada *data type* Makanan saat diletakkan pada kulkas. *Command* ini ekuivalen dengan (M).area.ySize.
  - dlvMkn(M)  
Selektor ini digunakan untuk menunjuk aspek waktu pengiriman atau pemrosesan pada *data type* Makanan. *Command* ini ekuivalen dengan (M).delivery.
- Konstruktor:
    - void CreateEmptyMakanan (Makanan \*mkn)  
I.S. Makanan tidak terdefinisi  
F.S. Makanan terdefinisi sebagai FoodMark  
Digunakan untuk mempersiapkan tipe bentukan Makanan sebelum dimasukkan data yang valid
  - Baca/ Tulis:
    - void printMakanan (Makanan mkn)  
I.S. Makanan terdefinisi, mungkin bisa saja merupakan FoodMark  
F.S. Makanan ditampilkan dengan format (ID) - (Nama makanan) - (waktu *delivery*) - <ukuran makanan> - (waktu *processing*) - (aksi yang dibutuhkan).  
Prosedur ini digunakan untuk menampilkan data makanan secara lengkap.
    - void printCatalog (Makanan mkn)  
I.S. Makanan terdefinisi, mungkin bisa saja merupakan FoodMark  
F.S. Makanan ditampilkan dengan format (Nama makanan) - (waktu *delivery*) - (aksi yang dibutuhkan) - (waktu *processing*)  
Prosedur ini digunakan untuk menampilkan makanan pada *catalog* (tidak semua data makanan ditampilkan)

- Fungsi/ Prosedur lain:
  - boolean isIDGreater(Makanan m1, Makanan m2)  
Mengirimkan true apabila ID m1 lebih besar dari ID m2  
Fungsi digunakan untuk membandingkan ID makanan ketika diurutkan
  - boolean isIDLower(Makanan m1, Makanan m2)  
Mengirimkan true apabila ID m1 lebih kecil dari ID m2  
Fungsi digunakan untuk membandingkan ID makanan ketika diurutkan

### 3.4 *Stack*

ADT stack terdiri dari tabel penyimpan elemen dan alamat TOP atau puncak seperti halnya yang kita tahu. ADT stack ini digunakan untuk undo dan redo yang merupakan salah satu fitur yang harus ada di mesin BNMO. Alasan menggunakan ADT stack ini karena bersesuaian dengan cara bekerja undo dan redo. Stack merupakan ADT yang menambahkan elemen ke paling atas dan mengeluarkan elemen paling atas juga secara berurutan. Sama halnya dengan alur kerja undo dan redo. ADT ini diimplementasikan dalam Stack di file stack.c.

- Selektor:
  - Top(S)
  - InfoTop(S)
- Konstruktor:
  - void CreateEmpty (Stack \* S)  
Membuat sebuah stack S yang kosong berkapasitas MaxEl.
- Fungsi/ Prosedur lain:
  - void Push (Stack \* S, state X)  
Menambahkan X sebagai elemen Stack S.
  - void Pop (Stack \*S, state \*X)  
Menghapus X dari Stack S.
  - void displayStack (Stack S)  
Untuk memperlihatkan Stack.
  - void Undo (Stack \*S\_undo, Stack \*S\_redo, state \*currentState, int totalcommand, POINT src)  
Melakukan Undo gerakan, mengembalikan simulator, waktu, dan peta sebelum.
  - void Redo (Stack \*S\_undo, Stack \*S\_redo, state \*currentState, int totalcommand, POINT src)  
Melakukan Redo gerakan, mengembalikan simulator, waktu, dan peta sesudah.

### 3.5 *Mesin Karakter dan Mesin Kata*

ADT Mesin Kata dan Mesin Karakter adalah ADT yang saling berkaitan, mesin karakter digunakan untuk mempermudah penggunaan mesin kata. Kedua ADT ini digunakan untuk membaca konfigurasi dari file. Alasan menggunakan ADT ini karena input konfigurasi terdiri dari baris-baris yang menandakan informasi yang berbeda-beda sehingga harus dibaca satu persatu yang sangat cocok dengan ADT Mesin Karakter. Selain itu, ADT ini juga digunakan dalam inisiasi serta membaca suatu input dari pengguna. ADT Mesin Kata memiliki tipe bentukan Word yang terdiri dari *array of char* sebagai penampung kata dan

*length* sebagai penanda panjang kata. ADT Mesin Karakter diimplementasikan dalam *charmachine.c* dan mesin kata pada *wordmachine.c*.

Dalam hal ini, ADT Mesin Kata telah diubah sedemikian rupa sehingga alih-alih membaca inputan karakter menggunakan *stdin* (*standard input*), ADT Mesin Kata digunakan untuk membaca sebuah string. Ketika meminta input dari user atau membaca konfigurasi file, program akan membaca setiap baris sebagai suatu string. String tersebut akan dibaca oleh ADT Mesin Kata secara satu per satu karakter.

### 3.5.1 Mesin Karakter

- Fungsi/ Prosedur lain
  - void START(char \*str, int \*idx)  
I.S. sembarang  
F.S. memulai pembacaan karakter pada string yang ingin dibaca dengan *currentChar* sebagai karakter pertama.
  - void ADV(int \*idx)  
I.S. *currentChar* adalah karakter yang sedang dibaca pada string  
F.S. memajukan satu karakter dalam pembacaan string sehingga *currentChar* adalah karakter berikutnya

### 3.5.2 Mesin Kata

- Selektor
  - len(w)  
Penunjuk panjang kata pada tipe bentukan Word.
  - str(w)  
Penunjuk *array of char* pada tipe bentukan Word.
- Fungsi/ Prosedur lain
  - void IgnoreBlanks(char \*str, int \*idx)  
I.S. CC sembarang  
F.S. CC tidak BLANK atau CC adalah MARK
  - void STARTWORD (char \*str, int \*idx)  
I.S. CC Sembarang  
F.S. EndWord = true dan CC = MARK atau EndWord = false
  - void ADVWORD (char \*str, int \*idx)  
I.S. CC adalah karakter pertama kata yang akan diakuisisi  
F.S. *currentWord* adalah kata terakhir yang sudah diakuisisi, bisa berakhir di BLANK atau di MARK
  - void CopyWord(char \*str, int \*idx)  
I.S. CC adalah karakter pertama dari kata  
F.S. *currentWord* berisi kata yang sudah diakuisisi, CC = BLANK atau CC = MARK
  - int WordToInt (Word word)  
Mengembalikan integer hasil konversi word ke integer
  - void printWord (Word word)  
I.S. word terdefinisi  
F.S. menampilkan kata yang ditampung pada word
  - boolean isWordEqual(Word word1, Word word2)

- Mengembalikan *true* apabila kedua word adalah berisikan karakter yang sama dan memiliki panjang yang sama
- boolean `isWordStringEqual(Word w, char*c)`  
Mengembalikan *true* apabila kata yang dikandung pada *c* berisikan karakter yang sama pada *array of char* yang dikandung oleh *w*
- boolean `isWordAllIntegers(Word w)`  
Mengembalikan *true* apabila semua karakter yang dikandung pada *array of char* pada *w* merupakan angka sehingga dapat diubah menjadi integer
- `Word appendWord(Word w1, Word w2)`  
Mengembalikan word yang merupakan penggabungan Word *w2* setelah Word *w1*

### 3.6 String

ADT string berisi kumpulan-kumpulan fungsi untuk mengelola string seperti mengubahnya ke dalam integer, mencari panjangnya dan lain-lain. ADT ini digunakan untuk mempermudah dalam mengelola inputan bertipe string. Alasan menggunakan ADT ini karena memang adt ini cukup mudah diimplementasikan serta ADT string ini banyak dibutuhkan terutama mengelola inputan dari pengguna di program utama. ADT ini diimplementasikan dalam file `string.c`.

- Konstruktor
  - `void createStringEmpty(char *str)`  
I.S. *str* terdefinisi  
F.S. membuat *str* menjadi string kosong atau bila string sudah terisi, maka dibuat menjadi kosong
- Fungsi/ Prosedur lain
  - `void upper(char *str)`  
I.S. *str* terdefinisi, bisa berisi huruf, angka, atau apapun  
F.S. setiap karakter huruf kecil pada *str* diubah menjadi karakter huruf kapital
  - boolean `isBlank(char c)`  
Mengembalikan *true* apabila *c* merupakan karakter BLANK (spasi)
  - int `MakeCharToInt(char c)`  
Mengembalikan nilai karakter dari karakter *c* menjadi sebuah integer
  - int `stringToInt (char* str)`  
Mengembalikan nilai dari semua karakter dalam *str* menjadi sebuah integer
  - int `lengthString(char *str)`  
Mengembalikan panjang dari sebuah string sebagai sebuah integer
  - boolean `isStringEqual (char *str1, char *str2)`  
Mengembalikan *true* apabila setiap karakter pada *str1* sama dengan karakter pada *str2* dengan panjang *str1* sama dengan panjang *str2*
  - `void appendChar (char *str, char c)`  
I.S. *str* terdefinisi, mungkin kosong  
F.S. panjang *str* bertambah 1 dengan karakter *c* ditambahkan sebagai karakter paling belakang *str*
  - `void appendString (char *str1, char *str2)`

I.S. str1 dan str2 terdefinisi, keduanya mungkin kosong

F.S. str2 ditambahkan di belakang str1 dan panjang str1 menjadi penjumlahan dari panjang str1 dan str2

- o void copyString (char \*str1, char \*str2)

I.S. str1 dan str2 terdefinisi

F.S. str2 menjadi sama seperti str1

### 3.7 Simulator

ADT simulator terdiri dari nama simulator, lokasi dari simulator, inventory, delivery, dan processlist. ADT ini digunakan untuk menyimpan data-data yang ada ke dalam satu simulator.

- Selektor :

- o Nama(S)
- o Lokasi(S)
- o Inventory(S)
- o Delivery(S)
- o ProcessList(S)

- Konstruktor :

- o void CreateSimulator (Simulator \*S, Word nama, POINT P, PrioQueueTime Q, PrioQueueTime D, PrioQueueTime PL)  
Membuat sebuah simulator yang terdiri dari nama simulator, lokasi simulator, inventory, delivery, dan processlist.

- Fungsi/Procedure lain :

- o void ReadSimulator (Simulator \*S)  
Membaca Simulator.
- o void DisplaySimulator (Simulator S)  
Menampilkan nama dan lokasi simulator.
- o void DisplayNama (Simulator S)  
Menampilkan nama simulator.
- o void DisplayLokasi (Simulator S)  
Menampilkan koordinat lokasi simulator.
- o void DisplayInventory (Simulator S)  
Menampilkan list inventory
- o void MixOlahInventory (PrioQueueTime \*Q, PrioQueueTime \*DestQ, Cookbook cb, ID id, int idx, ListStatik fs)  
Mengupdate isi inventory jika melakukan mix.
- o void ChopOlahInventory (PrioQueueTime \*Q, PrioQueueTime \*DestQ, Makanan X1, Makanan X2)  
Mengupdate isi inventory jika melakukan chop.
- o void FryOlahInventory (PrioQueueTime \*Q, PrioQueueTime \*DestQ, Cookbook cb, ID id, int idx, ListStatik fs)  
Mengupdate isi inventory jika melakukan fry.
- o void BoilOlahInventory (PrioQueueTime \*Q, PrioQueueTime \*DestQ, Cookbook cb, ID id, int idx, ListStatik fs)  
Mengupdate isi inventory jika melakukan boil.

- void RemoveMakanan(PrioQueueTime \*Q, Makanan M)  
DequeueAt makanan pada inventory
- void GeserLokasi (Simulator \*S, int arah)  
Mengubah koordinat lokasi simulator
- void CreateSimulatorUndo (Simulator \*S, Word nama, POINT P, PrioQueueTime invent, PrioQueueTime DeliveryList, PrioQueueTime ProcessList)  
Membuat Simulator dummy untuk melakukan alokasi ulang isi simulator supaya isi dari stack undo tidak terpengaruh oleh pengurangan waktu yang ada

### 3.8 Resep

ADT Resep merupakan turunan dari ADT tree yang digunakan untuk meyimpan konfigurasi resep, tetapi ADT resep berisi tipe data Cookbook yang merupakan array bersisi Tree (Resep). Program ini memiliki beberapa primitif yaitu:

- Selektor
  - NResep(c)
  - Buku(c)
  - Resep(c,i)
- Fungsi/ Prosedur lain
  - boolean canFry(PrioQueueTime pq)  
PiroQueueTime pq merupakan inventory, mengembalikan true jika terdapat minyak di inventory dan mengembalikan false jika tidak ada minyak di inventory
  - boolean canMake(Cookbook cb, Makanan m, PrioQueueTime pq)  
Cookbook cb, Makanan m dan PiroQueueTime pq terdefinisi, megembalikan true jika bahan-bahan makanan yang dibutuhkan untuk membuat makanan m terdapat pada inventory, dan sebaliknya.
  - void printResep(Cookbook cb, ListStatik foods)  
I.S. Cookbook cb dan ListStatik foods terdefinisi  
F.S. Mencetak semua resep yang ada pada cookbook dengan konfigurasi <NAMA> <ACTION> - <BAHAN 1> - <BAHAN 2> ...<BAHAN N>

### 3.9 ListStatik

ADT list statik adalah list yang didalamnya digunakan untuk menyimpan elemen bertipe Makanan. ADT list ini digunakan untuk mempermudah dalam penampilan suatu daftar makanan salah satunya pada catalog(salah satu fitur yang harus ada di mesin BNMO). Alasan menggunakan ADT ini karena pengaksesan elemen bertipe Makanan menjadi lebih mudah. ADT ini diimplementasikan dalam file liststatik.c.

- Selektor:
  - ELMTLIST(l, i)  
Mengirimkan isi dari list statik l pada indeks i.
- Konstruktor:
  - void CreateListStatik(ListStatik \*l)  
Membuat list statik kosong.
- Primitif:

- `boolean isMark(ElTypeList a)`  
Mengembalikan true jika a adalah mark, dan false jika tidak.
- `boolean different(ElTypeList a, ElTypeList b)`  
Mengembalikan true jika a dan b berbeda, false apabila a dan b sama.
- `Makanan getMakanan(ID id, ListStatik foods)`  
Mengembalikan makanan berdasarkan id-nya.
- `int listLength(ListStatik l)`  
Mengirimkan banyaknya elemen efektif pada liststatik
- `IdxType getFirstIdx(ListStatik l)`  
Mengirimkan indeks elemen l pertama
- `IdxType getLastIdx(ListStatik l)`  
Mengirimkan indeks elemen l terakhir
- `boolean isIdxValid(ListStatik l, IdxType i)`  
Mengirimkan true jika i adalah indeks yang valid untuk kapasitas List l yaitu antara indeks yang terdefinisi untuk container.
- `boolean isIdxEff(ListStatik l, IdxType i)`  
Mengirimkan true jika i adalah indeks yang terdefinisi untuk List l.
- `boolean isEmpty(ListStatik l)`  
Mengirimkan true jika list l kosong, mengirimkan false jika tidak.
- `boolean isFull(ListStatik l)`  
Mengirimkan true jika list l penuh, mengirimkan false jika tidak.
- `boolean isListEqual(ListStatik l1, ListStatik l2)`  
Mengirimkan true jika l1 sama dengan l2, mengirimkan false jika tidak.
- `int indexOf(ListStatik l, ElTypeList val)`  
Mengirimkan indeks val pada l jika val ditemukan di dalam l. Jika val tidak ditemukan di dalam l, mengirimkan -1.
- `void insertFirst(ListStatik *l, ElTypeList val)`  
Menambahkan val sebagai elemen pertama l
- `void insertAt(ListStatik *l, ElTypeList val, IdxType idx)`  
Menambahkan val pada l di indeks ke idx.
- `void insertLast(ListStatik *l, ElTypeList val)`  
Menambahkan val sebagai elemen terakhir l.
- `void deleteFirst(ListStatik *l, ElTypeList *val)`  
Menghapus elemen pertama l dan menyimpannya pada val.
- `void deleteAt(ListStatik *l, ElTypeList *val, IdxType idx)`  
Menghapus elemen pada indeks ke idx di dalam l dan menyimpannya ke dalam val.
- `void deleteLast(ListStatik *l, ElTypeList *val)`  
Menghapus elemen terakhir l dan menyimpannya pada val.
- `int searchIndexOlahMakanan(ListStatik l, char* command, int count)`  
Mengembalikan index dari makanan pada list makanan sesuai yang dibutuhkan oleh fungsi pengolahan makanan.



### 3.10 Tree

ADT tree adalah adt yang akan digunakan untuk penyimpanan resep. Data yang disimpan dalam tree adalah integer id, Address childs dan integer nChild. Program ini memiliki beberapa primitif, yaitu:

- Selektor :
  - o Parent(T)
  - o NChild(T)
  - o Childs(T)
  - o Child(T, i)
- Konstruktor :
  - o Adress newTree(ID id)  
Mengirimkan address dari alokasi sebuah elemen
  - o void makeBranch(NTree \*T, int nchild)  
I.S. Ntree dan nchild terdefinisi  
F.S. Alokasi child bejumlah nchild dengan elemen child Nil
  - o void CreateTree(NTree \*T)  
I.S. NTree terdefinisi  
F.S. Terbentuk NTree kosong, yaitu elemen child Nil
  - o void mergeTree(NTree \*T, NTree T1)  
I.S. NTree T dan NTree T1 terdefinisi  
F.S. Jika T1 merupakan child dari T, maka mengembalikan T yang diperpanjang dengan T1. Jika T1 bukan merupakan child T, tidak melakukan perpanjangan.
  - o void addChild(NTree \*T, ID id)  
I.S. T dan id terdefinisi  
F.S. Melakukan alokasi elemen dengan nilai id, menambahkan elemen pada NTree T sebagai child T
  - o void deallocTreeNode (Address p)  
I.S. p Terdefinisi  
F.S. p didealokasi/ dikembalikan ke sistem
  - o boolean isEmptyTree(NTree T)  
mengembalikan true jika T = Nil, atau mengembalikan false jika T != Nil
  - o boolean isOneElmt(NTree p)  
mengembalikan true jika semua child dari p bernilai Nil, atau mengembalikan false jika terdapat child dari p yang bernilai Nil
  - o boolean isChildOf(NTree T, ID id)  
mengembalikan true jika id merupakan child dari T, vice versa
  - o void printTreeLevel(NTree T, int h, int l)  
I.S. T terdefinisi  
F.S. Mencetak tree sesuai level secara rekursif
  - o void printTree(NTree T, int h)  
I.S. T terdefinisi  
F.S. Mencetak tree
  - o void checkMerge(NTree \*T, NTree B)  
I.S. T dan B terdefinisi

- F.S. Mengembalikan T yang diperpanjang dengan B, jika B merupakan child T. Proses dilakukan secara rekursif
- int depth(Address T)  
mengembalikan nilai depth/ kedalaman dari Tree T

### 3.11 Prioqueue

ADT Prioqueue adalah ADT Queue yang terurut membesar berdasarkan "time".

Prioqueue digunakan sebagai inventory, buy, dan delivery dengan data yang disimpan dalam queue adalah tipe data Makanan. Program ini memiliki beberapa primitif, yaitu:

- Selektor :
  - Head(Q)
  - Tail(Q)
  - InfoHead(Q)
  - InfoTail(Q)
  - MaxEl(Q)
  - Elmt(Q,i)
- Konstruktor :
  - boolean IsEmptyQueue (PrioQueueTime Q)  
Mengirim true jika Q kosong: lihat definisi di atas
  - boolean IsFullQueue (PrioQueueTime Q)  
Mengirim true jika tabel penampung elemen Q sudah penuh yaitu mengandung elemen sebanyak MaxEl
  - int NBElmt (PrioQueueTime Q)  
Mengirimkan banyaknya elemen queue. Mengirimkan 0 jika Q kosong.
  - void MakeEmptyQueue (PrioQueueTime \* Q, int Max)  
I.S. sembarang  
F.S. Sebuah Q kosong terbentuk dan salah satu kondisi sbb: Jika alokasi berhasil, Tabel memori dialokasi berukuran Max+1 atau : jika alokasi gagal, Q kosong dg MaxEl=0 Proses : Melakukan alokasi, membuat sebuah Q kosong
  - void DeAlokasi(PrioQueueTime \* Q)  
Proses: Mengembalikan memori Q  
I.S. Q pernah dialokasi  
F.S. Q menjadi tidak terdefinisi lagi, MaxEl(Q) diset 0
  - void EnqueueInventory (PrioQueueTime \* Q, Makanan M)  
Proses: Menambahkan X pada Q dengan aturan priority queue, terurut membesar berdasarkan expiry time  
I.S. Q mungkin kosong, tabel penampung elemen Q TIDAK penuh  
F.S. X disisipkan pada posisi yang tepat sesuai dengan prioritas, TAIL "maju" dengan mekanisme circular buffer
  - void EnqueueDelivery (PrioQueueTime \* Q, Makanan M)  
Proses: Menambahkan X pada Q dengan aturan priority queue, terurut membesar berdasarkan delivery time  
I.S. Q mungkin kosong, tabel penampung elemen Q TIDAK penuh

- F.S. X disisipkan pada posisi yang tepat sesuai dengan prioritas,  
TAIL "maju" dengan mekanisme circular buffer
- void Dequeue (PrioQueueTime \* Q, Makanan \* M)  
Proses: Menghapus X pada Q dengan aturan FIFO  
I.S. Q tidak mungkin kosong  
F.S. X = nilai elemen HEAD pd I.S., HEAD "maju" dengan mekanisme circular buffer  
Q mungkin kosong
  - void PrintPrioQueueTimeInventory (PrioQueueTime Q);  
Mencetak isi queue Q ke layar untuk menunjukkan isi inventory
  - void PrintPrioQueueTimeDelivery (PrioQueueTime Q)  
Mencetak isi queue Q ke layar untuk menunjukkan isi delivery
  - int PencariMakanan(PrioQueueTime \*Q, Makanan M)  
Menemukan index dari makanan pada prioqueue
  - void DequeueAt(PrioQueueTime \*Q, Makanan M, Makanan \*X)  
Mengeluarkan makanan tertentu pada prioqueue
  - void decrementNExp(PrioQueueTime \*Q, int N)  
Mengurangi setiap waktu expiry pada queue sebesar N menit
  - void incrementNExp(PrioQueueTime \*Q, int N)  
Menambahi setiap waktu expiry pada queue sebesar N menit
  - void decrementHMEExp(PrioQueueTime \*Q, int hours, int minutes)  
Mengurangi setiap waktu expiry pada queue sebesar 'hours' jam dan 'minutes' menit
  - void decrementNDEL(PrioQueueTime \*Q, int N)  
Mengurangi setiap waktu delivery pada queue sebesar N menit
  - void incrementNDEL(PrioQueueTime \*Q, int N)  
Menamabah setiap waktu delivery pada queue sebesar N menit
  - void decrementHMDel(PrioQueueTime \*Q, int hours, int minutes)  
Mengurangi setiap waktu delivery pada queue sebesar 'hours' jam dan 'minutes' menit
  - void DequeueAtIndex(PrioQueueTime \*Q, int idx, Makanan \*m)  
Mengeluarkan makanan dari PrioQueueTime Q dengan index tertentu

### 3.12 Matrix

ADT matrix terdiri dari baris dan kolom. ADT ini digunakan untuk menampilkan map untuk BNMO melakukan proses pengolahan terhadap makanan. Alasan menggunakan ADT ini karena ADT matrix paling mudah dan tidak rumit untuk menampilkan suatu data 2 dimensi dibandingkan menggunakan list itu justru kurang efisien dan cenderung lebih rumit lagi. ADT ini diimplementasikan dalam Matrix di file matrix.c.

- Selektor
  - ROW\_EFF(M)
  - COL\_EFF(M)
  - ELMTMAT(M, i, j)
- Konstruktor
  - void createMatrix(int nRows, int nCols, Matrix \*m);

Membentuk sebuah Matrix "kosong" yang siap diisi berukuran nRow x nCol di "ujung kiri" memori.

I.S. nRow dan nCol adalah valid untuk memori matriks yang dibuat.

F.S. Matriks m sesuai dengan definisi di atas terbentuk.

- Fungsi atau procedure lain

- o IdxType getLastIdxRow(Matrix m)

Mengirimkan Index baris terbesar m.

- o IdxType getLastIdxCol(Matrix m)

Mengirimkan Index kolom terbesar m.

- o boolean isMatIdxEff(Matrix m, IdxType i, IdxType j)

Mengirimkan true jika i, j adalah Index efektif bagi m.

- o void copyMatrix(Matrix mIn, Matrix \*mOut)

Melakukan assignment mOut <- mIn.

- o void displayMatrix(Matrix m)

I.S. m terdefinisi.

F.S. Nilai ELMTMAT(M, i, j) ditulis ke layar per baris per kolom, masing-masing elemen per baris dipisahkan sebuah spasi. Baris terakhir tidak diakhiri dengan newline.

Proses: Menulis nilai setiap elemen m ke layar dengan traversal per baris dan per kolom.

- o boolean isSquare(Matrix m)

Mengirimkan true jika m adalah matriks dg ukuran baris dan kolom sama.

- o boolean isSymmetric(Matrix m)

Mengirimkan true jika m adalah matriks simetri : isSquare(m) dan untuk setiap elemen m,  $m(i,j)=m(j,i)$ .

- o POINT searchCharInMatrix(Matrix m, char c)

Mengembalikan index posisi dari char C pada matrix m dalam bentuk POINT

- o boolean canSwap(Matrix m, POINT des)

mengembalikan true jika elemen matrix pada des adalah ' ' (artinya bisa bergerak/bukan obstacle).

- o void swapElmt(Matrix \* m, POINT \*src, POINT des)

I.S. m, src, des = terdefinisi

F.S. jika dapat ditukar (canSwap = true) elemen pada src bertukar dengan elemen pada des, jika tidak dapat ditukar maka tidak melakukan apa-apa.

## 4 Program Utama

Program utama terletak pada file main.c. Pada mulanya mesin akan mengeluarkan sebuah tampilan ascii art berisi permulaan dan pengenalan. Setelah itu, user dipaksa untuk memasukkan input START/start huruf besar atau kecil akan diterima untuk menjalankan mesin BNMO ini. Setelah command Start ini diinputkan maka program akan memposisikan simulator pada koordinat (3,1) program juga akan meminta set waktu yang diinginkan. Kemudian user menentukan pilihan. Pilihan yang ada adalah sebagai berikut.

- Move north untuk menggerakkan simulator ke petak sebelah atas

STEI- ITB	IF2110_TB_02_A	Halaman 20 dari 71 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.		

- Move east untuk menggerakkan simulator ke petak sebelah kanan
- Move south untuk menggerakkan simulator ke petak sebelah bawah
- Move west untuk menggerakkan simulator ke petak sebelah kiri
- Buy untuk proses membeli suatu makanan
- Mix untuk proses penggabungan suatu makanan
- Fry untuk proses penggorengan makanan
- Chop untuk proses pemotongan makanan
- Boil untuk proses perebusan makanan
- Catalog untuk melihat daftar list makanan yang valid
- Cookbook untuk melihat data list resep yang valid
- Inventory untuk membuka inventory yang dimiliki oleh simulator
- Delivery untuk membuka tampilan list makanan yang sedang dikirim
- Process untuk membuka tampilan list makanan yang sedang diproses
- Wait (x) (y) untuk mempercepat waktu selama x jam dan y menit
- Undo untuk mengembalikan kondisi sebelum terlaksananya suatu proses pengolahan
- Redo mengembalikan kembali action yang telah di undo
- Fridge show untuk menampilkan isi kulkas
- Fridge take untuk mengembalikan isi kulkas
- Fridge put untuk meletakkan makanan pada kulkas
- Help untuk melihat semua *command* yang valid selama pelaksanaan program
- Exit untuk keluar dari mesin

Mesin BNMO akan berjalan sesuai dengan input command yang diberikan hingga user menginputkan command exit. Jika exit maka program akan berhenti berjalan.

## 5 Algoritma-Algoritma Menarik

### 5.1 Kulkas

Keseluruhan dari algoritma kulkas lumayan menarik. Pertama-tama pembuatan ADT Kulkas didasari oleh ADT lainnya seperti Matrix dan Makanan. Dengan begitu, pembuatan ADT Kulkas terasa seperti implementasi dari ADT yang telah dibuat sebelumnya. Kedua, dalam pembuatan kulkas, perlu dipertimbangkan aspek *identifier* untuk setiap makanan. Hal ini diperlukan untuk membedakan suatu makanan dengan makanan lainnya. Dalam hal ini, jika terdapat dua makanan pada kulkas dan makanan tersebut berbeda, tentunya akan mudah bagi *user* untuk mengidentifikasi makanan apa yang diambil. Namun, bila terdapat dua makanan yang sejenis, lalu ditaruh bersebelahan, akan sulit bagi *user* untuk menentukan makanan yang mana yang perlu diambil. Dalam hal ini, diperlukan *identifier* untuk membantu *user* mengidentifikasi makanan yang ingin diambil. Bagian ini dianggap menarik karena pembuat program ditantang untuk melihat program yang akan dibuatnya dari sudut pandang *user*.

### 5.2 Mekanisme Penyesuaian Waktu pada PriorityQueue

Salah satu *command* yang mungkin sering digunakan saat program dijalankan adalah *command* WAIT. Hal ini dilakukan agar *user* tidak perlu menghabiskan waktu hanya untuk menunggu makanan tersebut datang. Namun, karena *command* WAIT berpengaruh

STEI- ITB	IF2110_TB_02_A	Halaman 21 dari 71 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.		

langsung terhadap waktu, tentu saja dampak yang diakibatkan oleh *command* tersebut juga berpengaruh terhadap makanan yang ada pada program, terkhusus kepada makanan yang berada di *list* dan di *inventory*.

Dalam penggunaan *command* WAIT, perlu dipertimbangkan seberapa lama WAIT tersebut dilakukan. Bila menunggu terlalu lama, bukan hal yang tidak mungkin untuk makanan yang sebelumnya masih berada di *delivery list* berikutnya langsung dianggap kadaluarsa setelah WAIT berhenti dijalankan. Lantas perlu dipertimbangkan adanya pengurangan terhadap waktu kadaluarsa sebelum makanan masuk ke Inventory. Contohnya sebagai berikut:

Asumsikan terdapat sebuah makanan yang akan sampai dalam kurun waktu 8 menit. Makanan tersebut memiliki waktu kadaluarsa 1 jam. Bila *user* memilih untuk melakukan *command* WAIT selama 10 menit, tentu saja makanan tersebut akan sampai dan masuk ke *inventory*. Namun, alih-alih waktu kadaluarsa makanan tersebut bernilai 1 jam, waktu kadaluarsa makanan tersebut akan menjadi 58 menit ketika dimasukkan ke dalam *inventory*. Hal ini diakibatkan selama 2 menit terakhir, makanan seharusnya sudah sampai. Dengan begitu, waktu kadaluarsa makanan yang sudah sampai sudah sewajarnya dikurangi oleh selisih durasi wait dengan durasi sisa pengiriman. Aspek ini dianggap menarik karena menantang pembuat program untuk memperhatikan bahkan detail-detail kecil pada program.

## 6 Data Test

### 6.1 Start, Konfigurasi, dan Main Menu

Fitur ini digunakan untuk memulai pelaksanaan program. Pertama-tama, *user* akan disambut dengan *ascii art* dan diminta untuk menginputkan 'start'. Input tersebut akan menjadi penanda sebagai dijalkannya program. Program tidak akan dijalankan bila program tidak menerima input 'start'.

Program lalu meminta input dari *user* sebagai nama dari simulator yang akan digunakan. Tidak ada batasan untuk input *user* dalam hal ini. Berikutnya, program akan meminta input dari *user* sebagai waktu awal dari program. Mengingat hal ini bukanlah bagian dari Tugas Besar, diasumsikan bahwa *user* selalu menginput nilai yang benar, yaitu berupa integer. Setelah itu, program akan membaca semua konfigurasi dari file yang ada pada folder Textfile. Program akan menampilkan bahwa semua konfigurasi telah selesai dimuat oleh program. Terakhir, program yang telah memuat semua konfigurasi akan menampilkan *main menu*.



6.2.1 Move North

Deskripsi: Posisi simulator sebelum move adalah pada titik (3, 1).

```
=====
===== MAIN MENU =====
=====
sim di posisi: (3,1)
Waktu: < Day 0 | 0:0 >
Notifikasi : -
* * * * *
*           *
*   S T     X   *
*   M       X   *
*           X   *
*       X X X X   *
*   X           *
*   X           C   *
*   X X X     F   *
*           *
*   K       B   *
* * * * *
```

Figure 3 Tampilan Simulator Sebelum Move North

Deskripsi: Setelah move north dilakukan, posisi simulator adalah (3, 0). Letak huruf ‘S’ pada peta disesuaikan dengan posisi simulator.

```
Command: move north
=====
===== MOVE =====
=====
Simulator telah bergerak ke arah yang diinginkan.
=====
===== MAIN MENU =====
=====
sim di posisi: (3,0)
Waktu: < Day 0 | 0:3 >
Notifikasi : -
* * * * *
*   S           *
*       T     X   *
*   M       X   *
*           X   *
*       X X X X   *
*   X           *
*   X           C   *
*   X X X     F   *
*           *
*   K       B   *
* * * * *
```

Figure 4 Tampilan Simulator Setelah Move North



6.2.2 Move South

Deskripsi: Posisi simulator sebelum move adalah pada titik (2, 3).

```
=====
===== MAIN MENU =====
=====
sim di posisi: (2,3)
Waktu: < Day 0 | 0:7 >
Notifikasi : -
* * * * *
*           *
*      T     X   *
*   M         X   *
*     S       X   *
*       XXXX   *
*   X         *
*   X           C  *
*  XXX       F    *
*           *
*   K         B    *
* * * * *
* * * * *
```

Figure 5 Tampilan Posisi Simulator Sebelum Move South

Deskripsi: Setelah move south dilakukan, posisi simulator adalah pada titik (2, 4). Letak huruf ‘S’ pada peta disesuaikan dengan posisi simulator.

```

Command: move south
=====
===== MOVE =====
=====
Simulator telah bergerak ke arah yang diinginkan.
=====
===== MAIN MENU =====
=====
sim di posisi: (2,4)
Waktu: < Day 0 | 0:8 >
Notifikasi : -
* * * * *
*           *
*       T   X   *
*   M       X   *
*           X   *
*   S   X X X X *
*   X           *
*   X           C *
*   X X X   F   *
*           *
*   K       B   *
* * * * *

```

Figure 6 Tampilan Simulator Setelah Move South

### 6.2.3 Move West

Deskripsi: Posisi simulator sebelum move adalah pada titik (3, 6).

```

=====
===== MAIN MENU =====
=====
sim di posisi: (3,6)
Waktu: < Day 0 | 0:13 >
Notifikasi : -
* * * * *
*           *
*       T   X   *
*   M       X   *
*           X   *
*       X X X X *
*   X           *
*   X S       C *
*   X X X   F   *
*           *
*   K       B   *
* * * * *

```

Figure 7 Tampilan Simulator Sebelum Move West

Deskripsi: Setelah move west dilakukan, posisi simulator adalah pada titik (2, 6).  
Letak huruf ‘S’ pada peta disesuaikan dengan posisi simulator.

```
Command: move west
=====
=====              MOVE              =====
=====
Simulator telah bergerak ke arah yang diinginkan.
=====
=====              MAIN MENU              =====
=====
sim di posisi: (2,6)
Waktu: < Day 0 | 0:14 >
Notifikasi : -
* * * * *
*           T       X       *
*      M           X       *
*           X       *
*      X X X X       *
*      X           *
*      X S           C       *
*      X X X       F       *
*           *
*      K           B       *
* * * * *
```

Figure 8 Tampilan Simulator Setelah Move West

6.2.4 Move East

Deskripsi: Posisi simulator sebelum move adalah pada titik (5, 6).

```
=====
=====              MAIN MENU              =====
=====
sim di posisi: (5,6)
Waktu: < Day 0 | 0:17 >
Notifikasi : -
* * * * *
*           T       X       *
*      M           X       *
*           X       *
*      X X X X       *
*      X           *
*      X       S   C       *
*      X X X       F       *
*           *
*      K           B       *
* * * * *
```

Figure 9 Tampilan Simulator Sebelum Move East

Letak huruf 'S' pada peta disesuaikan dengan posisi simulator.

```
Command: move east
=====
                         MOVE
=====
Simulator telah bergerak ke arah yang diinginkan.
=====
                         MAIN MENU
=====

sim di posisi: (6,6)
Waktu: < Day 0 | 0:18 >
Notifikasi : -
* * * * *
*           *
*       T   X   *
*    M         X   *
*              X   *
*        X X X X   *
*     X          *
*    X            S C *
*   X X X      F   *
*                *
*    K          B   *
* * * * *
```

Figure 10 Tampilan Simulator Setelah Move East

### 6.2.5 Invalid Move atau Invalid Input

Deskripsi: Posisi simulator sebelum move adalah pada titik (6,6).

```
=====
MAIN MENU
=====

sim di posisi: (6,6)
Waktu: < Day 0 | 0:18 >
Notifikasi : -
* * * * *
*
*           T       X       *
*      M           X       *
*           X       *
*      X X X X       *
*      X           *
*      X           S C   *
*      X X X       F     *
*
*      K           B     *
* * * * *

```

Figure 11 Tampilan Simulator Sebelum Move

Deskripsi: Setelah move dilakukan, posisi simulator tetap pada titik (6,6) karena titik tujuan move tidak valid. Pada titik tujuan move terdapat tempat melakukan chop (C) sehingga simulator tidak dipindahkan ke titik tersebut.

```
Command: move east
=====
===== MOVE =====
=====
Simulator tidak bisa masuk ke petak tersebut.
=====
===== MAIN MENU =====
=====
sim di posisi: (6,6)
Waktu: < Day 0 | 0:18 >
Notifikasi : -
* * * * *
*           *
*      T   X   *
*  M       X   *
*           X   *
*      X X X X  *
*  X           *
*  X           S C *
*  X X X       F   *
*           *
*  K           B   *
* * * * *
```

Figure 12 Tampilan Simulator Ketika Input Invalid

6.3 Buy

Fitur Buy digunakan untuk melakukan pembelian makanan dengan keterangan aksi adalah buy sehingga makanan yang ditampilkan pada list buy hanya makanan yang bisa dibeli. Untuk melakukan Buy, posisi simulator (S) harus berada tepat di sebelah (T), jika tidak program akan menganggap aksi Buy tidak valid dan tidak bisa dilakukan. Jika aksi Buy berhasil dilakukan, makanan yang baru dibeli akan dimasukkan ke dalam delivery list. Sebelum makanan dimasukkan ke delivery list, pengguna perlu memasukkan input integer nomor urut makanan yang ingin dibeli berdasarkan list buy. Input selain integer atau integer di luar angka pada list buy akan mengakibatkan invalid input.

Deskripsi: Posisi simulator sebelum melakukan buy adalah pada titik (3,1). Ditampilkan list makana yang bisa dibeli, kemudian dipilih makanan no. 3 (telur) untuk dibeli. Karena posisi dan input valid, buy berhasil dilakukan dan telur dimasukkan ke dalam delivery list.

```
Command: buy
=====
===== BUY =====
=====
List Bahan Makanan yang Bisa Dibuat:
1. Lalapan
2. Nasi Uduk
3. Telur
4. Minyak Goreng
5. Tepung
6. Air
7. Cabai
8. Bawang
9. Ayam Mentah
Ketik 0 untuk kembali ke Main Menu.
Command: 3
Telur berhasil dipesan. Makanan akan diantar dalam 15 menit
=====
===== MAIN MENU =====
=====
sim di posisi: (3,1)
Waktu: < Day 0 | 0:1 >
Notifikasi : -
* * * * *
*
*      S T      X      *
*    M          X      *
*          X      *
*        X X X X      *
*    X              *
*    X              C      *
*    X X X      F      *
*
*    K          B      *
* * * * *
```

Figure 13 Tampilan Fitur Buy

Deskripsi: Posisi simulator sebelum melakukan buy adalah pada titik (3,1). Ditampilkan list makana yang bisa dibeli. Meskipun posisi simulator valid, input pengguna tidak valid karena input yang diminta adalah nomor urut makanan yang ingin dibeli (integer), bukan nama makanan yang ingin dibeli. Karena input tidak valid, perintah buy tidak dilakukan.

```

Command: buy
=====
===== BUY =====
=====
List Bahan Makanan yang Bisa Dibuat:
1. Lalapan
2. Nasi Uduk
3. Telur
4. Minyak Goreng
5. Tepung
6. Air
7. Cabai
8. Bawang
9. Ayam Mentah
Ketik 0 untuk kembali ke Main Menu.
Command: Lalapan
Invalid input. Input bukanlah integer atau integer tersebut tidaklah valid.

```

Deskripsi: Posisi simulator sebelum melakukan buy adalah pada titik (2, 1). Karena posisi simulator untuk melakukan buy tidak valid, buy tidak akan dilakukan.

```

Command: buy
=====
===== BUY =====
=====
Simulator tidak bersebelahan dengan tempat melakukan BUY.
Pastikan Simulator berada di sebelah petak 'T'
=====
===== MAIN MENU =====
=====
sim di posisi: (2,1)
Waktu: < Day 0 | 0:2 >
Notifikasi : -
* * * * *
*           *
*   S   T   X   *
*   M           X   *
*           X   *
*       X X X X   *
*   X           *
*   X           C   *
*   X X X   F   *
*           *
*   K           B   *
* * * * *

```

Figure 15 Tampilan Fitur Buy

## 6.4 Mix

Fitur ini digunakan untuk mengolah makanan dengan cara “mix” yang ada pada *inventory simulator*, pengolahan akan dilakukan dengan menghapus bahan makanan yang

dibutuhkan yang ada di *inventory* dan menyimpan kembali hasil makanan yang dibuat ke dalam *inventory*.

```
a di posisi: (6,8)
Waktu: < Day 1 | 1:46 >
Notifikasi : -
* * * * *
*           T   X   *
*   M         X   *
*           X   *
*       X X X X   *
*   X           *
*   X           C   *
*   X X X       F   *
*           T S M   *
*   K           B   *
* * * * *

Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: mix
=====
===== MIX =====
=====
List Bahan Makanan yang Bisa Dibuat:
1. Nasi Uduk Telur Pecel Ayam
2. Ayam Goreng Extra Sambal
3. Ayam Tepung
4. Sambal
Ketik 0 untuk kembali ke Main Menu.
Command: 4
Sambal berhasil diproses. Makanan akan diproses dalam 30 menit
```

Figure 16 Tampilan Fitur Mix Proses Berhasil

```
a di posisi: (6,8)
Waktu: < Day 1 | 1:47 >
Notifikasi : -
* * * * *
*           T   X   *
*   M         X   *
*           X   *
*       X X X X   *
*   X           *
*   X           C   *
*   X X X       F   *
*           T S M   *
*   K           B   *
* * * * *

Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: mix
=====
===== MIX =====
=====
List Bahan Makanan yang Bisa Dibuat:
1. Nasi Uduk Telur Pecel Ayam
2. Ayam Goreng Extra Sambal
3. Ayam Tepung
4. Sambal
Ketik 0 untuk kembali ke Main Menu.
Command: 1
Kamu tidak punya bahannya
```

Figure 17 Tampilan Fitur Mix Saat Tidak ada Bahan



```

k di posisi: (6,8)
Waktu: < Day 1 | 1:24 >
Notifikasi : -
* * * * *
*
*      T      X
*      X
* M      X
*      X
*      X X X X
*      X
*      X      C
*      X X X
*      T S
*      K      C
* * * * *

Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: mix
=====
===== MIX =====
=====
Simulator tidak bersebelahan dengan tempat melakukan MIX.
Pastikan Simulator berada di sebelah petak 'M'

```

Figure 18 Tampilan Fitur Mix Saat Posisi Simulator Tidak Bersebelahan Dengan Tempat Mix

## 6.5 Fry

Fitur ini digunakan untuk mengolah makanan dengan cara "fry" yang ada pada *inventory* simulator. Pengolahan akan dilakukan dengan menghapus bahan makanan yang dibutuhkan yang ada di *inventory* dan menyimpan kembali hasil makananyang dibuat ke dalam *inventory*.

```
a di posisi: (6,8)
Waktu: < Day 1 | 2:16 >
Notifikasi :
1. Sambal sudah selesai diproses dan dimasukkan ke dalam inventory.
*****
*                               *
*                               X *
*       X X X X *
*               X *
*       X X X X *
*   X *
*   X *
* X X X   F *
*           T S M *
*   K       B *
*****
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: fry
=====
===== FRY =====
=====
List Bahan Makanan yang Bisa Dibuat:
1. Telur Goreng
2. Ayam Goreng
3. Sambal Goreng
Ketik 0 untuk kembali ke Main Menu.
Command: 3
Sambal Goreng berhasil diproses. Makanan akan diproses dalam 3
0 menit
```

Figure 19 Tampilan Fitur Fry Saat Berhasil

```
a di posisi: (6,8)
Waktu: < Day 1 | 2:17 >
Notifikasi : -
*****
*                               *
*       T   X *
*   M       X *
*           X *
*       X X X X *
*   X *
*   X *
* X X X   F *
*           T S M *
*   K       B *
*****
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: fry
=====
===== FRY =====
=====
List Bahan Makanan yang Bisa Dibuat:
1. Telur Goreng
2. Ayam Goreng
3. Sambal Goreng
Ketik 0 untuk kembali ke Main Menu.
Command: 1
Kamu tidak punya bahannya
```

Figure 20 Tampilan Fitur Fry Saat Tidak ada Bahan

```

=====
k di posisi: (6,8)
Waktu: < Day 1 | 1:24 >
Notifikasi : -
* * * * *
*
*      T      X
*      M      X
*      X      X
*      X X X X
*      X
*      X      C
*      X X X
*      T S
*      K      C
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: fry
=====
===== FRY =====
=====
Simulator tidak bersebelahan dengan tempat melakukan FRY.
Pastikan Simulator berada di sebelah petak 'F'
=====

```

Figure 21 Tampilan Fitur Fry Saat Simulator Tidak Bersebelahan dengan tempat Fry

## 6.6 Chop

Fitur ini digunakan untuk mengolah makanan dengan cara “*chop*” yang ada pada *inventory* simulator. Pengolahan akan dilakukan dengan menghapus bahan makanan yang dibutuhkan yang ada di *inventory* dan menyimpan kembali hasil makanan yang dibuat ke dalam *inventory*.

```

=====
k di posisi: (6,8)
Waktu: < Day 1 | 1:23 >
Notifikasi :
1. Ayam Mentah sudah diterima oleh BNMO!
* * * * *
*
*      T      X
*      M      X
*      X      X
*      X X X X
*      X
*      X      C
*      X X X
*      T S
*      K      C
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: chop
=====
===== CHOP =====
=====
List Bahan Makanan yang Bisa Dibuat:
1. Ayam Potong
Ketik 0 untuk kembali ke Main Menu.
Command: 1
Ayam Potong berhasil diproses. Makanan akan diproses dalam 1 jam
=====

```

Figure 22 Tampilan Fitur Chop Saat Berhasil

```

=====
k di posisi: (6,8)
Waktu: < Day 1 | 1:24 >
Notifikasi : -
* * * * *
*
*      T      X      *
*    M          X      *
*              X      *
*            X X X X      *
*    X              *
*    X              C      *
*  X X X              *
*              T S      *
*    K              C      *
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: chop
=====
===== CHOP =====
=====
List Bahan Makanan yang Bisa Dibuat:
1. Ayam Potong
Ketik 0 untuk kembali ke Main Menu.
Command: 1
Kamu tidak punya bahannya
=====

```

Figure 23 Tampilan Fitur Chop Saat Tidak ada Bahan

```

=====
k di posisi: (7,8)
Waktu: < Day 1 | 1:25 >
Notifikasi : -
* * * * *
*
*      T      X      *
*    M          X      *
*              X      *
*            X X X X      *
*    X              *
*    X              C      *
*  X X X              *
*              T S      *
*    K              C      *
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: chop
=====
===== CHOP =====
=====
Simulator tidak bersebelahan dengan tempat melakukan CHOP.
Pastikan Simulator berada di sebelah petak 'C'
=====

```

Figure 24 Tampilan Fitur Chop Saat Simulator tidak bersebelahan dengan tempat Chop

## 6.7 Boil

Fitur ini digunakan untuk mengolah makanan dengan cara "boil" yang ada pada *inventory* simulator. Pengolahan akan dilakukan dengan menghapus bahan makanan yang dibutuhkan yang ada di *inventory* dan menyimpan kembali hasil makananyang dibuat ke dalam *inventory*.

```
=====
a di posisi: (6,8)
Waktu: < Day 1 | 2:47 >
Notifikasi : -
* * * * *
*
*      T      X      *
*    M          X      *
*          X      *
*        X X X X      *
*    X          *
*    X          C      *
*  X X X      F      *
*          T S M      *
*    K          B      *
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: boil
=====
BOIL
=====
List Bahan Makanan yang Bisa Dibuat:
1. Telur Rebus
Ketik 0 untuk kembali ke Main Menu.
Command: 1
Telur Rebus berhasil diproses. Makanan akan diproses dalam 30 men
it
```

Figure 25 Tampilan Fitur Boil Saat Berhasil

```
=====
a di posisi: (6,8)
Waktu: < Day 1 | 2:48 >
Notifikasi : -
* * * * *
*
*      T      X      *
*    M          X      *
*          X      *
*        X X X X      *
*    X          *
*    X          C      *
*  X X X      F      *
*          T S M      *
*    K          B      *
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: boil
=====
BOIL
=====
List Bahan Makanan yang Bisa Dibuat:
1. Telur Rebus
Ketik 0 untuk kembali ke Main Menu.
Command: 1
Kamu tidak punya bahannya
```

Figure 26 Tampilan Fitur Boil Saat Tidak ada Bahan

```

k di posisi: (6,8)
Waktu: < Day 1 | 1:24 >
Notifikasi : -
* * * * *
*           *
*   T       *
*   M       *
*           *
*   X X X X *
*   X       *
*   X       *
*   X X X   *
*           *
*   T S     *
*   K       *
*           *
* * * * *

Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: boil
=====
BOIL
=====
Simulator tidak bersebelahan dengan tempat melakukan BOIL.
Pastikan Simulator berada di sebelah petak 'B'

```

Figure 27 Tampilan Fitur Boil Saat Tidak Bersebelahan dengan tempat boil

## 6.8 Catalog

Fitur ini digunakan untuk melihat semua makanan yang valid saat pembacaan konfigurasi makanan pada program. Data yang ditampilkan pada catalog adalah nama makanan, durasi kadaluarsa, aksi yang diperlukan untuk mendapatkan makanan tersebut, dan delivery atau process time. Untuk ukuran makanan pada kulkas dan ID makanan tidak ditampilkan pada pemanggilan fitur ini.

```

=====
CATALOG
=====
List Makanan
(Nama Makanan - Durasi Kadaluarsa - Aksi yang Diperlukan - Delivery atau Process Time)
1. Nasi Uduk Telur Pecel Ayam - 5 jam - MIX - 3 jam
2. Lalapan - 1 jam - BUY - 20 menit
3. Ayam Goreng Extra Sambal - 4 jam - MIX - 1 jam
4. Nasi Uduk - 2 jam - BUY - 30 menit
5. Telur Rebus - 1 jam 30 menit - BOIL - 30 menit
6. Telur Goreng - 1 jam - FRY - 20 menit
7. Ayam Goreng - 3 jam - FRY - 45 menit
8. Sambal Goreng - 2 jam 30 menit - FRY - 30 menit
9. Telur - 5 jam - BUY - 15 menit
10. Ayam Tepung - 1 jam 30 menit - MIX - 10 menit
11. Minyak Goreng - 4 jam - BUY - 45 menit
12. Sambal - 1 jam 30 menit - MIX - 30 menit
13. Ayam Potong - 1 jam - CHOP - 1 jam
14. Tepung - 45 menit - BUY - 20 menit
15. Air - 6 jam - BUY - 30 menit
16. Cabai - 1 jam 45 menit - BUY - 20 menit
17. Bawang - 3 jam 30 menit - BUY - 20 menit
18. Ayam Mentah - 1 jam - BUY - 20 menit

```

Figure 28 Tampilan Catalog yang Menampilkan Data dari Semua Makanan

## 6.9 Cookbook

Fitur ini digunakan untuk melihat semua resep pembuatan makanan yang valid saat pembacaan konfigurasi resep. Dalam fitur ini, ditampilkan semua makanan yang didapat melalui fitur pemrosesan makanan (apapun selain *buy*). Selain itu, ditampilkan juga aksi apa yang perlu dilakukan untuk mendapatkan makanan tersebut dan bahan makanan apa saja yang diperlukan.

```
=====
===== COOKBOOK =====
=====
List Resep
(aksi yang diperlukan - bahan...)
1. Nasi Uduk Telur Pecel Ayam
   MIX - Lalapan - Ayam Goreng Extra Sambal - Nasi Uduk - Telur Rebus
2. Ayam Goreng Extra Sambal
   MIX - Ayam Goreng - Sambal Goreng
3. Telur Rebus
   BOIL - Telur - Air
4. Telur Goreng
   FRY - Telur - Minyak Goreng
5. Ayam Goreng
   FRY - Ayam Tepung - Minyak Goreng
6. Sambal Goreng
   FRY - Minyak Goreng - Sambal
7. Ayam Tepung
   MIX - Ayam Potong - Tepung - Air
8. Ayam Potong
   CHOP - Ayam Mentah
9. Sambal
   MIX - Cabai - Bawang
```

Figure 29 Tampilan Cookbook yang Menampilkan Data dari Semua Resep Makanan

## 6.10 Inventory

Fitur *Inventory* merupakan fitur penyimpanan makanan yang dimiliki oleh simulator. Setiap makanan yang valid dan telah selesai dikirim atau diproses akan disimpan dalam *inventory* simulator. Makanan yang berada pada *inventory* memiliki waktu kadaluarsa. Waktu kadaluarsa ini akan terus berkurang selama simulator melakukan pekerjaan yang menghabiskan waktu. Makanan yang waktu kadaluarsanya sudah habis akan dibuang dari *inventory*. *Inventory* yang tidak kosong memiliki tampilan makanan yang terurut membesar berdasarkan waktu kadaluarsa.

```
=====
===== INVENTORY =====
=====
Tidak ada makanan pada inventory.
```

Figure 30 Tampilan Inventory Kosong

```

=====
===== INVENTORY =====
=====
List Makanan di Inventory:
(No - Nama - Waktu Sisa Kadaluwarsa)
1. Lalapan - 47 menit
2. Nasi Uduk - 1 jam 58 menit
3. Telur - 4 jam 44 menit

```

Figure 31 Tampilan Inventory Berisi 3 Makanan

## 6.11 Delivery

Fitur *Delivery* adalah fitur pengiriman makanan yang ditampilkan saat simulator memesan makanan menggunakan *command Buy*. Ketika simulator memesan makanan, makanan tidak akan langsung masuk ke dalam *inventory* milik simulator. Namun, makanan tersebut akan memiliki waktu pengiriman yang dapat dilihat pada *delivery list*. Jika simulator memesan lebih dari satu makanan, makanan yang ditampilkan pada *delivery list* akan mengurut membesar berdasarkan waktu pengiriman.

```

=====
===== DELIVERY =====
=====
Tidak ada makanan pada list delivery.

```

Figure 32 Tampilan Delivery List Kosong

```

=====
===== DELIVERY =====
=====
List Makanan di Delivery List:
No - Nama - Waktu Sisa Delivery
1. Bawang - 16 menit
2. Ayam Mentah - 18 menit
3. Air - 27 menit
4. Minyak Goreng - 44 menit

```

Figure 33 Tampilan Delivery List Berisi 4 Makanan

## 6.12 Process

Fitur *Process* adalah fitur yang merupakan pemenuhan untuk spesifikasi Bonus 2. Ketika simulator melakukan pemrosesan makanan, makanan yang diproses tidak akan langsung masuk ke dalam *inventory* milik simulator. Namun, makanan tersebut akan masuk terlebih dahulu ke dalam *list* yang disebut dengan *process list*. Jika simulator melakukan pemrosesan pada lebih dari satu makanan, makanan yang ditampilkan pada *process list* akan mengurut membesar berdasarkan waktu pemrosesan.



```
=====
===== PROCESS =====
=====
Tidak ada makanan pada list process.
```

Figure 34 Tampilan Process List Kosong

```
=====
===== PROCESS =====
=====
List Makanan di Process List:
No - Nama - Waktu Sisa Process
1. Sambal - 17 menit - MIX
2. Telur Goreng - 18 menit - FRY
3. Ayam Potong - 59 menit - CHOP
```

Figure 35 Tampilan Process List Berisi 3 Makanan

### 6.13 Wait (X)(Y)

Fitur Wait digunakan untuk melakukan penambahan waktu selama X jam dan Y menit. Wait dapat dilakukan di posisi mana pun. Input X dan Y adalah integer, X adalah lama waktu dalam jam, Y adalah lama waktu dalam menit. Jika Wait berhasil dilakukan, semua ADT yang memiliki keterangan waktu akan disesuaikan, seperti waktu simulator, sisa waktu delivery makanan, sisa waktu process makanan, dan sisa waktu kedaluwarsa makanan. Penambahan waktu selama 1 menit tidak akan dilakukan di akhir Wait, penambahan waktu Wait hanya sesuai dengan input X dan Y saja.

Deskripsi: Waktu simulator sebelum Wait adalah <Day 0 | 5:6>. Isi inventory saat sebelum

Wait adalah telur dengan waktu kedaluarsa 10 menit. Isi delivery list sebelum Wait adalah ayam mentah dengan waktu delivery 20 menit.

```
=====
===== MAIN MENU =====
=====
sim di posisi: (3,1)
Waktu: < Day 0 | 5:6 >
Notifikasi : -
* * * * *
*
*      S T      X      *
*    M          X      *
*          X      *
*        X X X X      *
*    X          *
*    X          C      *
*  X X X      F      *
*
*    K          B      *
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
```

Figure 36 Tampilan Wait

Deskripsi: Diberikan input Wait 0 30, program menambahkan 0 jam 30 menit pada waktu simulator. Waktu simulator setelah Wait adalah < Day 0 | 5:36 >. Setelah Wait dilakukan, telur sudah kedaluwarsa, ayam mentah sudah sampai dan dimasukkan ke dalam inventory.

```
Command: wait 0 30
=====
===== WAIT =====
=====
Menunggu 30 menit
Waktu pada Delivery List dan Inventory telah disesuaikan.
=====
===== MAIN MENU =====
=====
sim di posisi: (3,1)
Waktu: < Day 0 | 5:36 >
Notifikasi :
    1. Ayam Mentah sudah diterima oleh BNMO!
    2. Telur kedaluwarsa.. :(
* * * * *
*           *
*      S T   X   *
*    M       X   *
*           X   *
*        X X X X  *
*    X           *
*    X           C  *
*   X X X       F   *
*           *
*    K           B   *
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
```

Figure 37 Tampilan Wait

Deskripsi: Kasus ketika input X dan Y adalah 0, akan ditampilkan pesan berikut. Tidak terjadi perubahan apa pun terhadap kondisi simulator karena Wait tidak akan menambahkan 1 menit setelah aksi selesai, penambahan waktu hanya sesuai input X dan Y.

```

Command: wait 0 0
=====
===== WAIT =====
=====
Untuk apa dilakukan? Untuk apa? :(
=====
===== MAIN MENU =====
=====
sim di posisi: (3,1)
Waktu: < Day 0 | 5:36 >
Notifikasi : -
* * * * *
*
*      S T      X      *
*    M          X      *
*          X      *
*        X X X X      *
*    X          *
*    X          C      *
*  X X X      F      *
*
*    K          B      *
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.

```

Figure 38 Tampilan Wait

Deskripsi: Kasus ketika input X atau Y bukan integer, akan ditampilkan pesan berikut. Tidak terjadi perubahan apa pun pada terhadap kondisi simulator karena invalid input dan Wait tidak dilakukan.

```

Command: wait 30 menit
=====
=====          WAIT          =====
=====
Masukan tidak valid. Y bukan sebuah integer.
=====
=====          MAIN MENU          =====
=====
sim di posisi: (3,1)
Waktu: < Day 0 | 5:36 >
Notifikasi : -
* * * * *
*
*      S T      X      *
*      M          X      *
*              X      *
*          X X X X      *
*      X          *
*      X          C      *
*      X X X      F      *
*              *
*      K          B      *
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.

```

Figure 39 Tampilan Wait

### 6.14 Undo Redo

Undo berfungsi untuk membatalkan command yang sudah dilakukan dengan mengembalikan semua data ke sebelum command dilakukan. Di sisi lain, redo berfungsi untuk membatalkan undo. Tes yang dilakukan akan memperlihatkan nama simulator, lokasi simulator, peta, dan waktu simulator. Pada tes, akan dilakukan 1 kali MOVE SOUTH, lalu 1 kali UNDO, dan 1 kali REDO.

```

===== MAIN MENU =====
=====
ray di posisi: (3,1)
Waktu: < Day 10 | 20:30 >
Notifikasi : -
* * * * *
*
*      S T      X      *
*    M          X      *
*          X      *
*        X X X X      *
*    X          *
*    X          C      *
*  X X X      F      *
*
*    K          B      *
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.

```

Figure 40 Tampilan Undo Redo

Deskripsi awal:

- Posisi : (3,1)
- Waktu : Day 10 | 20:30
- Peta : Posisi S di sebelah kiri T

```

Command: move south
=====
===== MOVE =====
=====
Simulator telah bergerak ke arah yang diinginkan.
=====
===== MAIN MENU =====
=====
ray di posisi: (3,2)
Waktu: < Day 10 | 20:31 >
Notifikasi : -
* * * * *
*
*      T      X      *
*    M S      X      *
*          X      *
*        X X X X      *
*    X          *
*    X          C      *
*  X X X      F      *
*
*    K          B      *
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.

```

Figure 41 Tampilan Undo Redo

Deskripsi:

1. Command yang dilakukan: MOVE SOUTH
2. Hasil yang diinginkan:
  - a. Posisi: (3,2)  
Waktu: Day 10 | 20:31
  - b. Peta: Posisi S ke bawah 1
3. Hasil yang didapat:
  - a. Posisi: (3,2)  
Waktu: Day 10 | 20:31
  - b. Peta: Posisi S ke bawah 1

```

Command: undo
=====
                        UNDO
=====
Simulator telah bergerak ke arah yang diinginkan.
Undo telah dilakukan
=====
                        MAIN MENU
=====

ray di posisi: (3,1)
Waktu: < Day 10 | 20:30 >
Notifikasi :
    1. MOVE tidak jadi dilakukan.
* * * * *
*           *
*      S T   X   *
*    M       X   *
*           X   *
*        X X X X  *
*    X         *
*    X         C  *
*   X X X     F   *
*           *
*    K       B   *
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.

```

Figure 42 Tampilan Undo Redo

Deskripsi:

1. Command yang dilakukan: UNDO
2. Hasil yang diinginkan:
  - a. Posisi: (3,1)  
Waktu: Day 10 | 20:30
  - b. Peta: Posisi S kembali ke sebelah kiri T
3. Hasil yang didapat:
  - a. Posisi: (3,1)  
Waktu: Day 10 | 20:30
  - b. Peta: Posisi S kembali ke sebelah kiri T

```

Command: redo
=====
===== REDO =====
=====
Simulator telah bergerak ke arah yang diinginkan.
Redo telah dilakukan
=====
===== MAIN MENU =====
=====
ray di posisi: (3,2)
Waktu: < Day 10 | 20:31 >
Notifikasi :
  1. MOVE kembali dilakukan.
* * * * *
*           *
*      T   X   *
*  M  S     X   *
*           X   *
*      X X X X   *
*  X           *
*  X           C   *
*  X X X     F   *
*           *
*  K           B   *
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.

```

Figure 43 Tampilan Undo Redo

Deskripsi :

- Command yang dilakukan: REDO
- Hasil yang diinginkan:
  - Posisi: (3,2)
  - Waktu: Day 10 | 20:31
  - Peta: Posisi S kembali ke bawah 1
- Hasil yang didapat:
  - Posisi: (3,2)
  - Waktu: Day 10 | 20:31
  - Peta: Posisi S kembali ke bawah 1

### 6.15 Fridge

Fitur ini adalah fitur yang dibuat untuk pemenuhan Bonus 1. Pada dasarnya fitur ini terdiri dari 3 *commands* yaitu, FRIDGE SHOW, FRIDGE TAKE, dan FRIDGE PUT. Berdasarkan namanya, ketiga *commands* tersebut secara berturut-turut memiliki fungsi untuk menampilkan, mengambil, dan meletakkan makanan pada Kulkas.



### 6.15.1 Fridge Show

```
=====
FRIDGE
=====
Makanan ditulis dalam format < ID Makanan, Identifier >
Bari yang valid pada kulkas adalah 0..4 dan kolom yang valid pada kulkas adalah 0..9
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
```

Figure 44 Tampilan Command Fridge Show dan Kulkas Kosong

```
=====
FRIDGE
=====
Makanan ditulis dalam format < ID Makanan, Identifier >
Bari yang valid pada kulkas adalah 0..4 dan kolom yang valid pada kulkas adalah 0..9
[ <43,1> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <61,0> | <61,0> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <63,2> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <63,2> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <63,2> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
```

Figure 45 Tampilan Command Fridge Show dan Kulkas Memiliki Isi Kulkas

### 6.15.2 Fridge Take

```
=====
FRIDGE
=====
Kulkas kosong. Tidak ada yang bisa diambil dari kulkas.
```

Figure 46 Tampilan Command Fridge Take tetapi Kulkas Kosong

```
=====
FRIDGE
=====
Makanan ditulis dalam format < ID Makanan, Identifier >
Bari yang valid pada kulkas adalah 0..4 dan kolom yang valid pada kulkas adalah 0..9
[ <43,1> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <61,0> | <61,0> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <63,2> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <63,2> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <63,2> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
Masukkan petak yang untuk mengambil makanan dari kulkas: 4 1
Makanan Air telah diambil dari kulkas
```

Figure 47 Tampilan Command Fridge Take dengan Input Valid dan Berhasil

```
=====
FRIDGE
=====
Makanan ditulis dalam format < ID Makanan, Identifier >
Bari yang valid pada kulkas adalah 0..4 dan kolom yang valid pada kulkas adalah 0..9
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <11,0> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
Masukkan petak yang untuk mengambil makanan dari kulkas: 9 9
Tidak ada makanan pada petak tersebut.
```

```

=====
FRIDGE
=====
Makanan ditulis dalam format < ID Makanan, Identifier >
Bari yang valid pada kulkas adalah 0..4 dan kolom yang valid pada kulkas adalah 0..9
[ <43,1> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <61,0> | <61,0> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <63,2> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <63,2> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <63,2> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
Masukkan petak yang untuk mengambil makanan dari kulkas: 4 0
Tidak ada makanan pada petak tersebut.

```

Figure 48 Tampilan Command Fridge Take tetapi Input Tidak Valid

### 6.15.3 Fridge Put

```

=====
FRIDGE
=====
Tidak ada makanan pada inventory. Tidak ada yang bisa dimasukkan pada kulkas.

```

Figure 49 Tampilan Command Fridge Put tetapi Inventory Kosong

```

=====
FRIDGE
=====
Makanan ditulis dalam format < ID Makanan, Identifier >
Bari yang valid pada kulkas adalah 0..4 dan kolom yang valid pada kulkas adalah 0..9
[ <43,1> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | <61,0> | <61,0> | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
List Makanan di Inventory:
(No - Nama - Waktu Sisa Kadaluwarsa)
1. Nasi Uduk - 10 menit
2. Telur Goreng - 11 menit
3. Bawang - 2 jam 7 menit
4. Minyak Goreng - 3 jam
5. Air - 4 jam 43 menit
Masukkan makanan yang ingin dimasukkan : 5
Masukkan petak yang untuk menaruh makanan pada kulkas: 2 1
Makanan Air telah dimasukkan ke dalam kulkas

```

Figure 50 Tampilan Command Fridge Put dengan Input Valid dan Berhasil

```

=====
===== FRIDGE =====
=====
Makanan ditulis dalam format < ID Makanan, Identifier >
Bari yang valid pada kulkas adalah 0..4 dan kolom yang valid pada kulkas adalah 0..9
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
List Makanan di Inventory:
(No - Nama - Waktu Sisa Kadaluwarsa)
1. Nasi Uduk - 10 menit
2. Telur Goreng - 11 menit
3. Sambal - 40 menit
4. Ayam Potong - 52 menit
5. Bawang - 2 jam 7 menit
6. Minyak Goreng - 3 jam
7. Air - 4 jam 43 menit

Masukkan makanan yang ingin dimasukkan : 9
Input integer tidaklah valid.

```

```

=====
===== FRIDGE =====
=====
Makanan ditulis dalam format < ID Makanan, Identifier >
Bari yang valid pada kulkas adalah 0..4 dan kolom yang valid pada kulkas adalah 0..9
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
[ Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong | Kosong ]
List Makanan di Inventory:
(No - Nama - Waktu Sisa Kadaluwarsa)
1. Nasi Uduk - 10 menit
2. Telur Goreng - 11 menit
3. Sambal - 40 menit
4. Ayam Potong - 52 menit
5. Bawang - 2 jam 7 menit
6. Minyak Goreng - 3 jam
7. Air - 4 jam 43 menit

Masukkan makanan yang ingin dimasukkan : 3
Masukkan petak yang untuk menaruh makanan pada kulkas: 10 10
Makanan tidak bisa dimasukkan pada petak tersebut.

```

Figure 51 Tampilan Command Fridge Put tetapi Input Tidak Valid

#### 6.15.4 Invalid Input

```

=====
MAIN MENU
=====
a di posisi: (2,9)
Waktu: < Day 0 | 0:45 >
Notifikasi : -
* * * * *
*           T           X
*      M           X
*           X
*      X X X X
*           X
*      X           C
*      X X X       F
*           X
*      K S       B
* * * * *
Silahkan masukkan command yang ingin dilakukan.
Masukkan 'HELP' untuk melihat list command yang dapat digunakan.
Command: fridge haha
=====
FRIDGE
=====
Input selain 'SHOW', 'TAKE', dan 'PUT' tidak diterima.

```

Figure 52 Tampilan Command Fridge yang Invalid

## 7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Start, Konfigurasi, dan Main Menu	Mengetahui output yang dihasilkan Ketika program pertama kali berjalan	<ul style="list-style-type: none"> <li>Run Program</li> <li>Inputkan START/start</li> <li>Input nama simulator dan waktu awal</li> </ul>	START/start	Program dapat berjalan dan user dapat menginputkan command-command yang telah dibuat	Saat di run program menampilkan ascii art kemudian program meminta inputan dari user untuk memulai harus diinputkan START/start setelah itu user dapat menginputkan command-command yang tersedia.
2	Move	Mengetahui apakah simulator telah bergerak ke arah yang diinginkan user	1. Program sedang dijalankan	Move North	Lokasi simulator bergerak satu petak ke arah atas.	Lokasi: (3, 0)

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
			2. Melakukan command MOVE		Dalam tes digunakan: Lokasi: (3, 1)	
3	Buy	Mengetahui apakah barang yang dibeli masuk kedalam queue delivery dan masuk ke dalam inventory setelah beberapa waktu	<ol style="list-style-type: none"> <li>1. Program sedang dijalankan</li> <li>2. Menuju petak T</li> <li>3. Command buy</li> <li>4. Pilih nomor barang</li> <li>5. Cek delivery</li> <li>6. Cek inventory</li> </ol>	... Buy Delivery Inventory	Terjadi perubahan pada queue delivery dengan bertambahnya elemen sesuai barang yang dibeli, dan bertambahnya elemen setelah beberapa waktu	Queue delivery bertambah sesuai barang dan setelah beberapa menit masuk ke queue inventory
4	Wait	Mengetahui apakah terjadi perubahan waktu terhadap waktu simulator, inventory, delivery list, process list.	<ol style="list-style-type: none"> <li>1. Program sedang dijalankan.</li> <li>2. Cek inventory</li> <li>3. Cek delivery</li> <li>4. Cek process</li> <li>5. Melakukan command Wait X Y</li> <li>6. Cek inventory</li> <li>7. Cek delivery</li> <li>8. Cek process</li> </ol>	Wait X Y, X adalah lama jam, Y adalah lama menit.  Dalam tes ini digunakan Wait 0 30	<p>Terjadi perubahan waktu terhadap waktu simulator, inventory, delivery list, process list.</p> <p>Dalam tes ini digunakan: Waktu: Day 0   5:6. Inventory: Telur (kedaluarsa dalam 10 menit). Delivery list: ayam mentah (sampai dalam 19 menit). Process list: kosong.</p>	Waktu: Day 0   5:36 Inventory: Ayam mentah (kedaluarsa dalam 49 menit) Delivery List: kosong. Process list: kosong
5	UNDO	Mengetahui apakah user dapat membatalkan command.	<ul style="list-style-type: none"> <li>o Memulai program baru</li> <li>o Melakukan command yang mengubah data dan memajukan waktu (di sini MOVE SOUTH)</li> </ul>	UNDO	Posisi : (3,1) Waktu : Day 10   20:30 Peta : Posisi S kembali ke sebelah kiri T	Posisi : (3,1) Waktu : Day 10   20:30 Peta : Posisi S kembali ke sebelah kiri T

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
			<ul style="list-style-type: none"> <li>Melakukan command UNDO</li> </ul>			
6	REDO	Mengetahui apakah user dapat membatalkan undo.	<ul style="list-style-type: none"> <li>Memulai program baru</li> <li>Melakukan command yang mengubah data dan memajukan waktu (di sini MOVE SOUTH)</li> <li>Melakukan command UNDO</li> <li>Melakukan REDO</li> </ul>	REDO	Posisi : (3,2) Waktu : Day 10   20:31 Peta : Posisi S kembali ke bawah 1	Posisi : (3,2) Waktu : Day 10   20:31 Peta : Posisi S kembali ke bawah 1
7	Mix	Mengetahui apakah berhasil membuat makanan dengan metode <i>mix</i>	<ol style="list-style-type: none"> <li>Program sedang dijalankan</li> <li>Bahan makanan tersedia di inventory</li> <li>Menuju petak M</li> <li>Command mix</li> <li>Pilih nomor barang</li> <li>Cek process</li> <li>Command Wait X Y</li> <li>Cek inventory</li> </ol>	Mix Process Inventory	Program menghilangkan bahan dalam membuat makanan yang ada di inventory dan memasukan ke dalam queue proses. Setelah beberapa waktu memindahkan hasil makanan dari queue proses ke dalam queue inventory	Program menghilangkan bahan makanan di inventory dan memasukan hasil ke proses lalu ke inventory
8	Chop	Mengetahui apakah berhasil melakukan chop/pemotongan pada makanan	<ol style="list-style-type: none"> <li>Program sednag berjalan</li> <li>Bahan makanan tersedia</li> <li>Menuju petak C</li> <li>Command chop</li> <li>Pilih nomor barang</li> <li>Cek proses</li> <li>Command wait x y</li> <li>Cek inventory</li> </ol>	Chop Process Inventory	Program menghilangkan bahan dalam membuat makanan yang ada di inventory dan memasukan ke dalam queue proses. Setelah beberapa waktu memindahkan hasil makanan dari queue proses ke dalam queue inventory	Program menghilangkan bahan makanan di inventory dan memasukan hasil ke proses lalu ke inventory

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
9	Fry	Mengetahui apakah berhasil melakukan fry/menggoreng pada makanan	<ol style="list-style-type: none"> <li>1. Program sednag berjalan</li> <li>2. Bahan makanan tersedia</li> <li>3. Menuju petak F</li> <li>4. Command fry</li> <li>5. Pilih nomor barang</li> <li>6. Cek proses</li> <li>7. Command wait x y</li> <li>8. Cek inventory</li> </ol>	Fry Process Inventory	Program menghilangkan bahan dalam membuat makanan yang ada di inventory dan memasukan ke dalam queue proses. Setelah beberapa waktu memindahkan hasil makanan dari queue proses ke dalam queue inventory	Program menghilangkan bahan makanan di inventory dan memasukan hasil ke proses lalu ke inventory
10	Boil	Mengetahui apakah berhasil melakukan boil/merebus pada makanan	<ol style="list-style-type: none"> <li>1. Program sednag berjalan</li> <li>2. Bahan makanan tersedia</li> <li>3. Menuju petak B</li> <li>4. Command Boil</li> <li>5. Pilih nomor barang</li> <li>6. Cek proses</li> <li>7. Command wait x y</li> <li>8. Cek inventory</li> </ol>	Boil Process Inventory	Program menghilangkan bahan dalam membuat makanan yang ada di inventory dan memasukan ke dalam queue proses. Setelah beberapa waktu memindahkan hasil makanan dari queue proses ke dalam queue inventory	Program menghilangkan bahan makanan di inventory dan memasukan hasil ke proses lalu ke inventory
11	Catalog	Mengetahui makanan apakah semua konfigurasi makanan berhasil terbaca	<ol style="list-style-type: none"> <li>1. Program sedang berjalan</li> <li>2. Command Catalog</li> </ol>	Catalog	Program menampilkan catalog list	Ditampilkan semua catalog
12	Delivery	Mengetahui apakah simulator menyimpan delivery list sesuai aksi yang dilakukan.	<ol style="list-style-type: none"> <li>1. Program sedang dijalankan</li> <li>2. Jika ingin menambahkan isi delivery list, lakukan</li> </ol>	DELIVERY / delivery	<p>Program menampilkan delivery list simulator.</p> <p>Dalam tes ini digunakan:</p>	Ditampilkan delivery list: Bawang (16 menit), ayam mentah (18 menit), air (27 menit), minyak (44 menit).

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
			Buy terlebih dahulu. 3. Melakukan command Delivery		Simulator membeli bawang, ayam mentah, air, minyak.	
13	Process	Mengetahui apakah simulator menyimpan process list sesuai aksi yang dilakukan	1. Program sedang dijalankan 2. Jika ingin menambahkan isi process list, lakukan command Mix, Chop, Boil, atau Fry terlebih dahulu. 3. Melakukan command Process	PROCESS / process	Program menampilkan process list simulator.  Dalam tes ini digunakan: Simulator melakukan MIX, FRY, CHOP.	Ditampilkan process list: Sambal (17 menit, MIX), telur goreng (18 menit, FRY), ayam potong (59 menit, CHOP).
14	Inventory	Mengetahui apa saja makanan yang dimiliki oleh simulator termasuk makanan yang telah selesai dikirim atau telah selesai diproses. Mengetahui waktu kadaluwarsa yang tersisa pada setiap makanan.	1. Program sedang dijalankan. 2. Jika ingin melihat list makanan yang ada pada inventory lakukan command 'Inventory' 3. Menampilkan semua list makanan yang dimiliki dengan format (nama makanan – waktu kadaluwarsa).	Inventory	Program menampilkan list makanan yang dimiliki dan yang belum kadaluwarsa (waktu kadaluwarsa > 0).	Program menampilkan semua makanan terurut membesar berdasarkan waktu kadaluwarsa yang tersisa dengan format “nama makanan – waktu kadaluwarsa”
15	Cookbook	Mengetahui makanan apakah semua konfigurasi resep berhasil terbaca	1. Program sedang berjalan 2. Command cookbook	Cookbook	Program menampilkan cookbook list	Ditampilkan semua isi cookbook
16	Fridge show	Mengetahui semua makanan yang ada dalam fridge	1. Program sedang berjalan 2. Command “fridge SHOW”	Fridge SHOW	Program menampilkan semua isi kulkas/fridge.	Ditampilkan semua isi kulkas dengan format tabel yang setiap sel menampilkan format <ID,



No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
						identifier>. Apabila kosong akan menampilkan <kosong>
17	Fridge put	Menaruh sebuah makanan yang ada pada inventory ke petak kosong yang ada pada kulkas	<ol style="list-style-type: none"> <li>1. Program sedang berjalan</li> <li>2. Command “Fridge PUT”</li> <li>3. Melihat isi inventory, apabila tidak kosong lanjut ke langkah 4.</li> <li>4. Memasukkan nomor makanan berdasarkan yang ditampilkan pada inventory</li> <li>5. Memasukkan petak yang akan diisi pada kulkas dengan format “baris kolom”</li> <li>6. Memasukkan makanan berdasarkan nomor urut makanan dan koordinat petak yang telah dimasukkan.</li> </ol>	Fridge PUT 5 2 1	Karena input koordinat dan nomor makanan sudah valid diharapkan program akan menampilkan pesan bahwa makanan telah berhasil dimasukkan pada kulkas.	Ditampilkan pesan “Makanan Air telah dimasukkan ke dalam kulkas.”
18	Fridge Take	Mengambil sebuah item makanan dari kulkas apabila kulkas tidak kosong.	<ol style="list-style-type: none"> <li>1. Program sedang berjalan</li> <li>2. Command “Fridge take”</li> <li>3. Melihat isi kulkas, apabila kulkas tidak kosong, lanjut ke langkah 4.</li> <li>4. Program meminta koordinat dari petak kulkas yang ingin</li> </ol>	Fridge Take 2 1	Command dan koordinat sudah valid dan diasumsikan petak <4, 1> terisi makanan jadi akan ditampilkan pesan bahwa makanan tersebut telah berhasil diambil dari kulkas.	Ditampilkan pesan “Makanan Air telah diambil dari kulkas”

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
			diambil makanannya dengan format “Baris Kolom” 5. Apabila baris dan kolom valid dan ada makanan pada petak tersebut makan makanan yang ada pada petak tersebut akan diambil dari kulkas.			

## 8 Pembagian Kerja dalam Kelompok

No.	NIM	Nama	Tugas pada Source Code	Tugas pada Laporan
1.	13521109	Rizky Abdillah Rasyid	<ul style="list-style-type: none"> <li>○ ADT point</li> <li>○ ADT Priority Queue</li> <li>○ ADT N-ary Tree</li> <li>○ ADT Resep</li> <li>○ Pembacaan konfigurasi peta</li> <li>○ Pembacaan konfigurasi resep</li> <li>○ Mekanisme command move, pengolahan makanan (mix, chop, fry dan boil) dan cookbook</li> </ul>	<ul style="list-style-type: none"> <li>○ Struktur data ADT point</li> <li>○ Struktur data ADT Prioqueue</li> <li>○ Struktur data ADT N-ary Tree</li> <li>○ Struktur data ADT Resep</li> </ul>
2.	13521115	Shelma Salsabila	1. ADT Time 2. ADT Matriks 3. Commad Catalog 4. ASCII ART	5. Struktur data ADT Time 6. Struktur data ADT Matrix 7. Laporan bagian 1 8. Laporan bagian 9
3.	13521116	Juan Christopher Santoso	<ul style="list-style-type: none"> <li>○ Pembacaan Konfigurasi Makanan</li> </ul>	<ul style="list-style-type: none"> <li>○ Penjelasan Tambahan Spesifikasi</li> </ul>

			<ul style="list-style-type: none"> <li>○ ADT Makanan</li> <li>○ ADT Mesin Kata dan Mesin Karakter</li> <li>○ ADT String</li> <li>○ Bonus 1 (Kulkas)</li> <li>○ Bonus 2 (Pengolahan Makanan)</li> <li>○ Mekanisme Pengaturan Masuk, Keluar, dan Waktu pada Delivery List, Process List, dan Inventory pada Main</li> <li>○ Command Help</li> <li>○ Error Handling pada User Input</li> </ul>	Tugas (Bonus 1 dan 2) <ul style="list-style-type: none"> <li>○ Struktur Data ADT Makanan</li> <li>○ Struktur Data ADT Mesin Kata dan Mesin Kata</li> <li>○ Struktur Data ADT String</li> <li>○ Data Test Start, Konfigurasi, dan Main Menu</li> <li>○ Data Test Catalog</li> <li>○ Data Test Cookbook</li> <li>○ Data Test Inventory</li> <li>○ Data Test Delivery</li> <li>○ Data Test Process</li> <li>○ Data Test Fridge</li> </ul>
4.	13521118	Ahmad Ghulam Ilham	<ul style="list-style-type: none"> <li>• Penulisan notifikasi pada main</li> </ul>	<ul style="list-style-type: none"> <li>• Data Test Move</li> <li>• Data Test Buy</li> <li>• Data Test Wait</li> </ul>
5.	13521143	Raynard Tanadi	<ul style="list-style-type: none"> <li>○ ADT Stack</li> <li>○ ADT Simulator</li> <li>○ Command Undo, Redo</li> </ul>	<ul style="list-style-type: none"> <li>○ Struktur Data ADT Simulator</li> <li>○ Struktur Data ADT Stack</li> <li>○ Data Test Undo, Redo</li> <li>○ Test Script Undo, Redo</li> </ul>

6.	13521146	Muhammad Zaki Amanullah	<ul style="list-style-type: none"> <li>• ADT List Statik</li> <li>• Command Inventory</li> <li>• Command Wait</li> </ul>	<ul style="list-style-type: none"> <li>• Struktur Data ADT List Statik</li> <li>• Test Script Inventory.</li> <li>• Test Script Fridge Put</li> <li>• Test Script Fridge Show</li> <li>• Test Script Fridge Take</li> </ul>
----	----------	-------------------------	--	---

## 9 Lampiran

### 9.1 Deskripsi Tugas Besar 2

*“BNMO sedang memasak mengikuti program simulasi yang telah direkam Doni”*

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

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

### 9.2 Notulen Rapat

Telah dilaksanakan dua kali rapat dan dua kali asistensi.

#### 1. Notulensi rapat pertama

#### Pembahasan Tubes 1

**Meet Pertama** : 21/10/2022

**Anggota yang hadir** :

1. Rizky Abdillah Rasyid
2. Shelma Salsabila
3. Juan Christopher Santoso
4. Raynard Tanadi
5. Muhammad Zaki Amanullah

**Pembahasan** :

STEI- ITB	IF2110_TB_02_A	Halaman 60 dari 71 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.		

- Pembagian ADT  
Ketentuan : Boleh ngambil dari praktikum  
**Ada 7 ADT yang utama jangan lupa driver nya juga :**

- ADT point : Rizky
- ADT time : shelma
- ADT makanan : Juan
- ADT simulator : Raynard
- ADT List statik : zaki
- ADT matrix : shelma
- ADT queue : rizky
- ADT stack : raynard
- ADT mesin kata : Juan
- ADT tree : zaki

1. Konfigurasi

1. Makanan : Juan
2. Resep : Zaki
3. Peta : Rizki
4. Inisiasi : Juan
5. Ascii art : shelma

1) Asistensi Pertama

Kamis malam jam 08.00 online

1. Untuk tugas jangan lupa deadline nya Selasa malam 23.59

2. Notulensi Rapat Kedua

**Pembagian Tugas**

**Meet kedua : 31/10/2022**

**Anggota yang hadir :**

- b. Rizky Abdillah Rasyid
- c. Shelma Salsabila
- d. Juan Christopher Santoso
- e. Ahmad Ghulam Ilham
- f. Raynard Tanadi
- g. Muhammad Zaki Amanullah

**Pembahasan**

1. Pembahasan mengenai ADT tree, selektor ELMT ada di matriks dan list. Solusinya untuk elemen misal di list atau di matriks tambahkan contoh ElemenMat.
2. Mengenai stack UNDO dan REDO udah namun isi stack belum mencakup time dan sebagainya tapi untuk UNDO REDO aman.
3. Pembagian tugas di main
  - 1) Move : Rizky
  - 2) Buy : Ghulam
  - 3) Mix : Shelma
  - 4) Chop : Juan
  - 5) Fry : Juan
  - 6) Catalog : Shelma
  - 7) Cookbook : Rizky
  - 8) Inventory : Zaki

- 9) Delivery : Ghulam
- 10) Wait : Zaki
- 11) Undo : Raynard
- 12) Redo : Raynard

Bonus

Kulkas : Juan

**Pertanyaan :**

- 1. Simulator itu start di (0,0) atau engga
- 2. Simulator harus di declare di map atau ga

Daftar State buat UNDO REDO

- c. Peta Matrix (termasuk koordinat simulator)
- d. Waktu
- e. Inventory

**DEADLINE Jum'at 23.59**

- 3. Notulensi Asistensi pertama dan kedua  
Urutan lampiran dari form asistensi kedua kemudian ke asistensi pertama.

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

No. Kelompok/Kelas : Kelompok A/K2  
 Nama Kelompok :  
 Anggota Kelompok (Nama/NIM) :  
     1. Rizky Abdillah Rasyid (13521109)  
     2. Shelma Salsabila (13521115)  
     3. Juan Christopher Santoso (13521116)  
     4. Ahmad Ghulam Ilham (13521118)  
     5. Raynard Tanadi (13521143)  
     6. Muhammad Zaki Amanullah (13521146)

Asisten Pembimbing : Fabian Savero Diaz Pranoto (13519140)

---

## Asistensi II

<b>Tanggal : 01 November 2022</b>	<b>Catatan Asistensi: Terlampir di bawah</b>
<b>Tempat : Labpro</b>	

**Kehadiran Anggota Kelompok:**

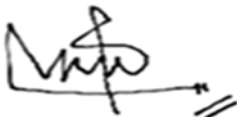
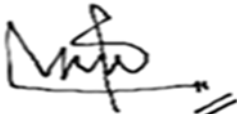
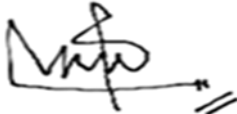
No  
NIM  
Tanda tangan

1



13521109

2



13521115

3



13521116

4



13521118

**Q :** Progress kelompok kalian?

**A :** Command parser semua udah jadi kecuali konfig makanan ada beberapa masalah yang baru diberesin tadi tapi udah aman. Untuk progress lain cukup aman. Progress kita minggu ini maju ke main.

**Q :** Untuk ADT tree kalian gimana?

**A :** ADT tree n array untuk anak-anaknya dipindahin ke list dinamik. Sekaligus dilakuin implementasi.

**Q :** Untuk undo ada maksimalnya ga?

**A :** Buat aja maximal stack nya berapa?

**Q :** Bonus kulkas ukuran makanan kita nentuin sendiri?

**A :** Iya

**Q :** Bonus fry, cook gak langsung selesai , itu simulator bisa pindah2 di tinggal?

**A :** iya

**Q :** Misal di undo barang di inventory balik lagi waktunya berkurang?

**A :** Iya



5



13521143

6




13521146

Berikut dilampirkan progress

Tanggal		01-11-2022
No	Fitur	Progress (0-100%)
1.	Command Parser	100%
2.	Inisiasi	
	a. Splash Screen	100%
	b. Command START	100%
	c. Command EXIT	100%
3.	Simulator	
	a. ADT Simulator	80%
4.	Makanan	
	b. Membaca makanan dari file	100%
	1. ADT Makanan	100%

	b. Command CATALOG	-
6.	Peta	
	a. Membaca peta dari file	100%
	• Command MOVE NORTH/EAST/SOUTH/WEST	100%
7.	Mekanisme Waktu	
	a. ADT Waktu	100%
	1. Waktu bertambah seiring command yg valid	100%
8.	Laporan (50%)	-

	<b>Tanda Tangan Asisten :</b> 
--	--

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

No. Kelompok/Kelas : Kelompok A/K2  
Nama Kelompok :  
Anggota Kelompok (Nama/NIM) :  
1. Rizky Abdillah Rasyid (13521109)  
2. Shelma Salsabila (13521115)  
3. Juan Christopher Santoso (13521116)  
4. Ahmad Ghulam Ilham (1352118)  
5. Raynard Tanadi (13521143)  
6. Muhammad Zaki Amanullah (13521146)

Asisten Pembimbing : Fabian Savero Diaz Pranoto (13519140)

---

### Asistensi I

Tanggal : 27 Oktober 2022	Catatan Asistensi: Terlampir di bawah
Tempat : zoom meeting	

**Kehadiran Anggota Kelompok:**

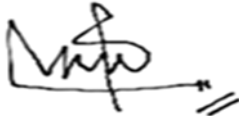
No  
NIM  
Tanda tangan

1



13521109

2



13521115

3



13521116

4



13521118

5



13521143

6

**Q (Kak Fabian) :** Sejauh mana pengerjaan tugas kalian?

**A :** Udah bikin beberapa ADT sama konfigurasi yang udah beres makanan dan map cuma resep belum jadi. ADT Tree juga masih on progress. Untuk Ascii Art udah ada tapi belum digabungin di main.

**Q :** Untuk resep, data-data makanan yang diperlukan cukup banyak, hanya saja untuk konfigurasi makanan baru ada 6. Tambahan data-data untuk makanan di resep itu dibikin sendiri? Aturan untuk pembuatannya gimana?

**A :** Itu bikin sendiri dibebasin mau bikin isi konfigurasinya gimana untuk id dan hal lainnya dibebasin ke kalian.

**Q :** Cukup penasaran sama union, union itu apa?

**A :** Union itu digunai biar kalian bisa nyimpen tipe yang berbeda. Misal dalam satu structure ada tipe char dan float.

**Q :** Jadi union itu bentuknya array?

**A :** Union bentuknya structure

**Q :** Union bisa membantu mempermudah, menurut kakak lebih efektif mana bikin union untuk makanan dan resep, atau bikin list masing-masing dari resep dan makanan itu.

**A :** Sebenarnya ini terserah tapi enaknya itu bisa nyimpen makanan dan resep dalam satu gitu jadi menurutku union cukup membantu.

**Q :** Penggunaan ADT Stack untuk UNDO dan REDO.


**A :** Kalian bisa nyimpen stack diprogram enaknya kalian tinggal pop atau push namun gak enaknya jadi ribet kalian ke kondisi awalnya. Nanti, dilaporan jelasin UNDO dan Redo gunain apa teknisnya kaya gimana.

**Q :** Word machine itu bisa diubah dari baca startinput jadi startstring. Jadi alih-alih menggunakan stdin justru baca string itu aman gak?

**A :** Gak papa itu Namanya ngeiterasi.



13521146

	<p><b>Q</b> : Untuk proses-proses kaya MIX itu masuk ke data type makanan. status actionnya masuk kemana?</p> <p><b>A</b> : Action ada di adt makanan kaya BUY, MIX, FRY gak ada ADT lain untuk menyimpan action lain selain makanan.</p> <p><b>O</b> : Anggap Auto BNMO gak ada dulu</p> <p><b>Q</b> : Bonus yang paling memungkinkan untuk dikerjain</p> <p><b>A</b> : Kulkas , waktu pengolahan makanan juga masih bisa dikerjain. Tapi kerjain wajibnya dulu baru kerjain bonus.</p> <p><b>Q</b> : Waktu pengolahan makanan itu maksudnya gimana?</p> <p><b>A</b> : Jadi setiap pengolahan diubah waktu. Adapun untuk makanan yang udah jadi mau langsung dimasukin ke inventory atau engga itu terserah kalian</p> <p><b>Q</b> : Untuk kulkas ukuran makanan berbeda-beda yang nentuin ukurannya kita</p> <p><b>A</b> : Iya terserah kalian</p> <p><b>Q</b> : Inventory buat kulkas itu bentuknya matrix kalau ukurannya 1x1 boleh ga?</p> <p><b>A</b> : boleh</p> <p><b>Q</b> : Form asistensi perlu tanda tangan semua anggota?</p> <p><b>A</b> : Iya, digital aja ya kirimnya</p>
	<p><b>Tanda Tangan Asisten :</b></p> 

### 9.3 Log Activity Anggota Kelompok

LOG ACTIVITY		
Tanggal	Nama	Deskripsi
21 Oktober 2022	Semua anggota	Rapat pertama pembahasan mengenai pembagian tugas membuat ADT yang diperlukan untuk membuat mesin
27 Oktober 2022	Semua anggota	Asistensi pertama menjelaskan tentang progress serta bertanya beberapa hal tentang kejelasan fitur pada asisten
31 Oktober 2022	Semua anggota	Rapat kedua membahas pembagian tugas untuk main
01 November 2022	Semua anggota	Asistensi kedua membahas mengenai beberapa progress serta pertanyaan-pertanyaan mengenai bonus