

LAPORAN TUGAS BESAR
IF2111 Algoritma dan Struktur Data STI

“PURRMART”


Dipersiapkan oleh:

Kelompok 6

Izhar Alif Akbar / 18223129
Harfhan Ikhtiar Ahmad R. / 18223123
Stefany Josefina Santono / 18223116
Nakeisha Valya Shakila / 18223133
Sharon Darma Putra / 18223107
Anggita Najmi Layali / 18223122

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		IF2111-TB-K3-6		39
		Revisi	-	20/12/2024

Daftar Isi

1 Ringkasan	4
2 Penjelasan Tambahan Spesifikasi Tugas	5
2.1 Store List Gacor (Bonus)	5
2.2 Spesifikasi Fitur Tambahan 2	6
2.3 Spesifikasi Fitur Tambahan 3	6
3 Struktur Data (ADT)	6
3.1 ADT Kustom	6
3.2 ADT List	7
3.3 ADT List Dinamis	8
3.4 ADT Mesin Karakter	9
3.5 ADT Mesin Kata	9
3.6 ADT Mesin Kalimat	10
3.7 ADT Queue	10
3.8 ADT Stack	11
3.9 ADT Setmap	12
3.10 ADT Linked List	12
4 Program Utama	13
5 Algoritma-Algoritma Menarik	21
5.1 Read Integer Name Items di Cart Remove	21
5.2 <Algoritma 2>	22
6 Data Test	22
6.1 START	22
Gambar 6.1.1 Menampilkan START yang valid	23
6.2 LOAD	23
Gambar 6.2.1 Menampilkan LOAD dengan file konfigurasi yang valid	23
6.3 SAVE	23
Gambar 6.3.1 Menampilkan SAVE yang valid	24
Gambar 6.3.2 Menampilkan SAVE yang tidak valid	24
6.4 STORE LIST	24
Gambar 6.4.1 Menampilkan STORE LIST	24
6.5 PROFILE	24
Gambar 6.5.1 Menampilkan PROFILE dengan Wishlist yang kosong	25
Gambar 6.5.1 Menampilkan PROFILE dengan keranjang kosong	25
6.6 CART ADD <nama> <n>	25
Gambar 6.6.1 Menampilkan CART ADD yang valid	25
6.7 CART REMOVE <nama> <n>	25

Gambar 6.7.1 Menampilkan CART REMOVE dengan barang tidak berada di keranjang	26
Gambar 6.7.3 Menampilkan CART REMOVE yang	26
dengan jumlah barang kurang dari jumlah di keranjang	26
6.8 CART SHOW	26
6.9 CART PAY	27
Gambar 6.9.1 Menampilkan CART PAY apabila saldo cukup	27
Gambar 6.9.2 Menampilkan CART PAY apabila dibatalkan	27
6.10 HISTORY <n>	27
6.11 WISHLIST ADD	28
Gambar 6.11.1 Menampilkan WISHLIST ADD dengan barang yang valid	28
Gambar 6.11.3 Menampilkan WISHLIST ADD	28
dengan barang sudah ada di wishlist	28
6.12 WISHLIST SWAP <i> <j>	28
Gambar 6.12.1 Menampilkan WISHLIST SWAP dengan wishlist kosong	29
Gambar 6.12.3 Menampilkan WISHLIST SWAP yang valid	29
6.13 WISHLIST REMOVE	29
Gambar 6.13.1 Menampilkan WISHLIST REMOVE apabila barang pada wishlist kosong	29
Gambar 6.13.3 Menampilkan WISHLIST REMOVE yang valid	30
6.14 WISHLIST REMOVE <i>	30
Gambar 6.14.1 Menampilkan WISHLIST REMOVE dengan wishlist kosong	30
Gambar 6.14.2 Menampilkan WISHLIST REMOVE yang valid	30
Gambar 6.14.3 Menampilkan WISHLIST REMOVE yang posisi ke-<i> tidak ditemukan	30
Gambar 6.14.4 Menampilkan WISHLIST REMOVE yang tidak memasukkan input angka	30
6.15 WISHLIST CLEAR	30
Gambar 6.15.1 Menampilkan WISHLIST CLEAR	31
6.16 WISHLIST SHOW	31
7 Test Script	31
8 Pembagian Kerja dalam Kelompok	34
9 Lampiran	35
9.1 Deskripsi Tugas Besar	35
9.2 Notulen Rapat	38
9.3 Log Activity Anggota Kelompok	40

1 Ringkasan

PURRMART adalah aplikasi berbasis command-line interface (CLI) yang dirancang untuk menggantikan sistem operasional manual di toko Borma, pemasok strategis OWCA, guna mendukung kebutuhan logistik dalam menghadapi ancaman besar. Aplikasi ini menawarkan solusi e-commerce yang interaktif dan efisien, meliputi pengelolaan barang toko, keranjang belanja, riwayat pembelian, dan wishlist pengguna. Dibangun menggunakan bahasa pemrograman C, aplikasi ini memanfaatkan struktur data seperti **Setmap**, **Stack**, dan **Linked List** untuk mengelola data secara efisien.

Laporan ini mencakup penjelasan lengkap mengenai berbagai aspek pengembangan aplikasi. Pendahuluan membahas latar belakang, tujuan, dan cakupan proyek, diikuti dengan spesifikasi teknis fitur utama. Penjelasan tentang struktur data yang digunakan menjelaskan bagaimana **Setmap** dimanfaatkan untuk keranjang belanja, **Stack** untuk riwayat pembelian, dan **Linked List** untuk wishlist, memastikan performa yang optimal. Selanjutnya, laporan ini juga menjelaskan implementasi program, termasuk menu utama dan perintah yang dapat digunakan, serta pengujian fungsionalitas melalui data dan skrip uji. Dokumentasi pengembangan yang mencakup pembagian tugas tim, dokumentasi rapat, dan asistensi juga disertakan untuk memberikan gambaran menyeluruh mengenai proses kerja.

Hasil Tugas Besar ini menunjukkan bahwa aplikasi PURRMART telah berhasil memenuhi spesifikasi yang diberikan. Semua fitur utama telah terimplementasi dengan baik, dan aplikasi mampu mensimulasikan proses e-commerce dengan akurat dan efisien. Antarmuka aplikasi dirancang agar mudah digunakan, memberikan pengalaman simulasi yang mendidik dan interaktif. Pengembangan proyek ini tidak hanya menghasilkan aplikasi yang fungsional, tetapi juga meningkatkan kemampuan tim dalam menyelesaikan proyek pemrograman yang kompleks secara kolaboratif dan terstruktur.

2 Penjelasan Tambahan Spesifikasi Tugas

2.1 Store List Gacor (Bonus)

Fitur **Store List Gacor** memanfaatkan ADT **ArrayDin** untuk menyimpan dan mengelola data barang secara dinamis. ADT ini mendukung operasi dasar seperti pengecekan daftar kosong (**IsEmpty**), penambahan elemen (**InsertLast**), dan iterasi melalui atribut **Neff** (jumlah elemen efektif). Dengan kompleksitas akses data **O(1)**, **ArrayDin** memungkinkan iterasi yang cepat dan efisien untuk menampilkan daftar barang dalam format tabel, memastikan bahwa hanya barang unik yang dicetak. Fleksibilitas **ArrayDin** juga memungkinkan pengelolaan barang di toko tanpa batasan kapasitas tetap.

Dibandingkan dengan ADT lain, seperti **Linked List** atau **Set**, **ArrayDin** menawarkan kombinasi efisiensi dan kesederhanaan yang sangat cocok untuk fitur ini. **Linked List** kurang efisien untuk akses langsung, sementara **Set** memerlukan logika

tambahan untuk menyimpan pasangan nama barang dan harga. Oleh karena itu, **ArrayDin** menjadi pilihan ideal untuk memenuhi spesifikasi **Store List Gacor**, memastikan barang ditampilkan dengan format yang rapi dan sesuai dengan kebutuhan toko.

3 Struktur Data (ADT)

Pada program PURRMART, kami menambahkan 3 jenis struktur data (ADT) sehingga terdapat 10 ADT. Berikut merupakan beberapa ADT yang digunakan dalam program ini:

3.1 ADT Kustom

SKETSA STRUKTUR DATA :

```
/* Kamus Umum */
#define MAX_LEN 100

#include "../LinkedList/linkedlist.h"
#include "../Stack/stack.h"
#include "../Setmap/setmap.h"

typedef struct {
    char name[MAX_LEN];
    char password[MAX_LEN];
    int money;
    Map keranjang;
    Stack riwayat_pembelian;
    Linkedlist wishList;
} User;

#endif
```

user.h

```
/* Kamus Umum */
#define MAX_LEN 100

typedef struct {
    char name[MAX_LEN];
    int price;
} Barang;

#endif
```

barang.h

Dalam program Purrmart, kami menggunakan ADT Kustom untuk merepresentasikan dua entitas, yaitu **User** dan **Barang**. Entitas **User** dirancang dengan tiga

atribut yaitu *name* berupa string untuk menyimpan nama pengguna, *password* berupa string untuk kata sandi pengguna, *money* berupa integer untuk mencatat jumlah uang yang dimiliki oleh pengguna, *keranjang* berupa tipe data Map untuk menyimpan barang yang akan dibeli, *riwayat pembelian* berupa tipe data Stack untuk mencatat riwayat barang yang dibeli, dan *wishList* berupa tipe data Linkedlist untuk mencatat barang yang menjadi wishlist user.. Sementara itu, Entitas **Barang** memiliki dua atribut utama, yaitu *name* berupa string untuk nama barang dan *price* berupa integer untuk harga barang. Dengan struktur ini, kami dapat mengelola data pengguna dan barang secara spesifik, yang memungkinkan kami untuk mengimplementasikannya pada command yang digunakan.

Kami mengimplementasikan kedua ADT ini dalam file bertipe header *user.h* dan *barang.h*. Pada *user.h*, ADT **User** didefinisikan sebagai struct dengan atribut *name*, *password*, dan *money*. Sedangkan pada *barang.h*, ADT **Barang** didefinisikan sebagai struct dengan atribut *name* dan *price*. Penggunaan kedua entitas tersebut digunakan untuk memisahkan setiap bagian program berdasarkan fungsinya.

3.2 ADT List

SKETSA STRUKTUR DATA :

```
/* Definisi ukuran maksimum elemen dalam list */
#define MaxEl 100

/* Definisi indeks dan elemen list */
typedef int IdxType;
typedef User ElType;

/* Definisi Mark untuk elemen kosong */
#define Mark -1

/* Definisi struktur list */
typedef struct {
    ElType A[MaxEl];
} List;
```

list.h

Dalam program Purrmart, kami menggunakan ADT List Statis untuk mengelola data pengguna yang bersifat tetap. ADT ini memanfaatkan *boolean.h* untuk operasi logika dan *user.h* untuk mendefinisikan tipe data User yang digunakan sebagai elemen dalam List Statis. ADT ini dirancang untuk menyimpan data secara efisien dengan ukuran tetap, sehingga cocok untuk data yang tidak berubah, seperti daftar pengguna yang terdaftar. Operasi yang didukung meliputi penambahan, penghapusan, pencarian elemen, dan penggabungan dua list. Dengan struktur array statis, ADT ini memastikan pengelolaan data yang cepat dan terorganisasi tanpa memerlukan pengelolaan memori tambahan.

ADT List Statis ini diimplementasikan dalam file header *list.h*. Struktur data utamanya terdiri dari atribut A, yang merupakan array statis berkapasitas maksimum MaxEl sebanyak 100 elemen. Elemen list direpresentasikan dengan ElType, yang bertipe User, sedangkan indeks array direpresentasikan dengan IdxType, bertipe integer. Selain itu,

list memiliki operasi pendukung seperti pencetakan elemen ke layar dan pencarian indeks elemen, menjadikan ADT ini ideal untuk kebutuhan data tetap dengan pengelolaan sederhana.

3.3 ADT List Dinamis

SKETSA STRUKTUR DATA :

```
// Boolean
#define boolean unsigned char
#define true 1
#define false 0

#define InitialSize 10

typedef int IdxType;
typedef Barang ElType2;
typedef struct {
    ElType2 *A;
    int Capacity;
    int Neff;
} ArrayDin;
```

arraydin.h

Dalam program Purrmart, kami juga menggunakan ADT List Dinamis untuk mengelola data barang secara fleksibel. ADT ini memanfaatkan boolean.h sebagai dasar operasi logika dan barang.h untuk mendefinisikan tipe data Barang yang digunakan sebagai elemen dalam List Dinamis. ADT ini dirancang untuk mendukung pengelolaan data yang sering berubah, seperti daftar barang yang dimasukkan ke dalam keranjang belanja. Fitur yang disediakan mencakup penambahan, penghapusan elemen, dan penyesuaian ukuran array secara dinamis, sehingga kapasitasnya dapat bertambah atau berkurang sesuai kebutuhan program.

ADT List Dinamis diimplementasikan dalam file header arraydin.h. Struktur data utamanya terdiri dari atribut A, yaitu pointer ke array dinamis bertipe ElType2, Capacity untuk mencatat kapasitas maksimum array saat ini, dan Neff untuk mencatat jumlah elemen yang efektif. Elemen list direpresentasikan oleh Barang, sedangkan indeks elemen menggunakan IdxType, bertipe integer. Dengan fleksibilitas ini, ADT List Dinamis sangat cocok untuk data yang bersifat dinamis, memungkinkan pengelolaan memori yang efisien tanpa batasan kapasitas tetap.

3.4 ADT Mesin Karakter

SKETSA STRUKTUR DATA :

```
#define MARK '\n'
/* State Mesin */
extern char currentChar;
```

```
extern boolean EOP;
```

mesinkarakter.h

Dalam program Purrmart, kami menggunakan ADT Mesin Karakter untuk membaca teks karakter demi karakter dari suatu pita, baik itu input langsung dari pengguna melalui stdin atau dari file. ADT Mesin Karakter ini berada pada file bertipe header *mesinkarakter.h*. Di dalamnya, terdapat dua state yaitu **currentChar** digunakan untuk merepresentasikan karakter yang sedang dibaca yang memiliki tipe data berupa character, dan **EOP (End of Process)**, digunakan untuk menunjukkan apakah pembacaan telah mencapai akhir pita yang memiliki tipe data boolean. Struktur pendukung lainnya yaitu **pita**, yang menjadi pointer ke sumber data (file atau standar input), dan **retval**, variabel untuk memantau hasil operasi pembacaan.

3.5 ADT Mesin Kata

SKETSA STRUKTUR DATA :

```
typedef struct
{
    char TabWord[NMax]; /* container penyimpan kata, indeks yang dipakai
    [0..NMax-1] */
    int Length;
} Word;

/* State Mesin Kata */
extern boolean endWord;
extern Word currentWord;
```

mesinkata.h

Dalam program Purrmart, kami menggunakan ADT Mesin Kata digunakan untuk membaca teks pada satu kata demi satu kata dengan memanfaatkan **MesinKarakter** sebagai dasar untuk membaca per karakter. Mesin Kata juga dapat dimanfaatkan dalam pengolahan teks dengan memisahkan kata-kata dari input, seperti mengabaikan spasi dan mengenali akhir input seperti MARK atau '!'. Selain itu, pada ADT Mesin Kata digunakan dalam melakukan *converting* tipe data tertentu agar sesuai dengan spesifikasi.

ADT Mesin Kata ini berada pada file bertipe header *mesinkarakter.h*. Di dalamnya, terdapat struktur data yang terdiri dari **Word**, yang memiliki atribut array bertipe character (TabWord) untuk menyimpan kata hingga panjang maksimum NMax, dan atribut Length bertipe integer untuk mencatat panjang kata yang sedang diproses. Selain itu, terdapat state berupa EndWord bertipe boolean yang menunjukkan apakah pembacaan telah mencapai akhir input, serta currentWord bertipe Word yang menyimpan kata yang sedang diproses.

3.6 ADT Mesin Kalimat

SKETSA STRUKTUR DATA :


```

#define NMaks 100
#define NEWLINE '\n'
#define MARK2 '\0'

typedef struct {
    char TabLine[NMaks+1];
    int Length;
} Kalimat;

extern boolean EndKalimat; // Deklarasi variabel global
extern Kalimat CLine;
extern Kalimat CInput;
extern Kalimat CCommand;
extern boolean endWord;

```

mesinkalimat.h

Dalam program Purrmart, kami menggunakan ADT Mesin Kalimat digunakan untuk membaca, memproses, dan menyimpan data teks berbasis baris. ADT ini memanfaatkan MesinKata sebagai dasar pemrosesan setiap kata-nya, yang dibuat dengan tujuan untuk mempermudah proses dalam melakukan load dan save suatu file config karena mampu membaca teks dari file ataupun input terminal, hingga menulis kembali teks ke file. Selain itu, ADT ini berfungsi untuk memisahkan bagian-bagian dalam kalimat, seperti angka atau kata, menggunakan fungsi parsing

ADT Mesin Kata ini berada pada file bertipe header *mesinkalimat.h*. Di dalamnya, terdapat struktur data yang terdiri dari Kalimat, yang memiliki atribut array bertipe character (TabLine) untuk menyimpan teks dan atribut Length bertipe integer untuk mencatat panjang teks tersebut. Selain itu, terdapat state berupa EndKalimat bertipe boolean yang menandakan apakah pembacaan kalimat sudah selesai atau belum, CLine bertipe Kalimat yang menyimpan kalimat yang sedang diproses, CInput bertipe Kalimat yang menginput kalimat dari pengguna, dan Command bertipe Kalimat yang memproses perintah yang ada.

3.7 ADT Queue

SKETSA STRUKTUR DATA :

```

#define IDX_UNDEF -1
#define CAPACITY 100
/* Definisi elemen dan address */
typedef Barang ElType2;
typedef struct
{
    ElType2 buffer[CAPACITY];
    int idxHead;
    int idxTail;
} Queue;

/* ***** AKSES (Selektor) ***** */

```

```

/* Jika q adalah Queue, maka akses elemen : */
#define IDX_HEAD(q) (q).idxHead
#define IDX_TAIL(q) (q).idxTail
#define HEAD(q) (q).buffer[(q).idxHead]
#define TAIL(q) (q).buffer[(q).idxTail]

```

queue.h

Dalam program Purrmart, kami menggunakan ADT Queue digunakan untuk memproses kalimat atau input dalam program dengan menggunakan konsep **Queue** (antrian). Queue berfungsi untuk menampung elemen kalimat (seperti kata atau simbol) secara berurutan dan diproses mengikuti prinsip First In and First Out, di mana elemen pertama yang masuk adalah yang pertama kali keluar. Selain itu, ADT ini juga mampu berfungsi untuk menambahkan elemen ke antrian, menghapus elemen, serta mengecek apakah antrian kosong atau penuh .

ADT Mesin Kata ini berada pada file bertipe header mesinkalimat.h. Di dalamnya, terdapat dua struktur data yang terdiri dari Eltype2 bertipe Barang. Dan Queue yang memiliki buffer untuk menyimpan elemen-elemen di dalam antrian bertipe ElType2, dan array bertipe integer dengan dua indeks, yaitu idxHead untuk menandakan elemen pertama dalam antrian dan idxTail untuk menandakan elemen terakhir.

3.8 ADT Stack

SKETSA STRUKTUR DATA :

```

#define MaxEl 100

typedef Barang infotype; // Change to the appropriate type for stack elements

typedef struct {
    infotype *T; // Dynamically allocated array for stack elements
    int TOP;     // Index of the top element in the stack
} Stack;

```

stack.h

Dalam program Purrmart, kami menggunakan ADT Stack untuk mengelola data transaksi atau aktivitas yang dilakukan pelanggan secara berurutan dengan prinsip LIFO (Last In, First Out). ADT ini memanfaatkan boolean.h sebagai dasar untuk operasi logika dan barang.h untuk mendefinisikan tipe data Barang yang digunakan sebagai elemen stack. ADT ini dibuat untuk menyimpan data sementara, seperti riwayat barang yang terakhir ditambahkan atau transaksi yang terakhir dilakukan, sehingga mempermudah pembatalan atau pengelolaan aktivitas tertentu. Selain itu, ADT ini berfungsi untuk mengelola data secara efisien menggunakan array dinamis, yang memungkinkan pengolahan data stack tanpa batasan ukuran tetap.

ADT Stack ini berada pada file header stack.h. Di dalamnya, terdapat struktur data Stack, yang memiliki atribut *T* berupa array dinamis untuk menyimpan elemen stack, serta *TOP* untuk menunjuk indeks elemen teratas. Struktur ini mendukung berbagai operasi

seperti menambahkan elemen ke stack, menghapus elemen teratas, dan menampilkan isi stack.

3.9 ADT Setmap

SKETSA STRUKTUR DATA :

```
#define MaxEl 100

typedef struct {
    Barang Key;    // Key is a Barang type
    int Quantity;  // Quantity of the item in the cart
} item;

typedef struct {
    item *Elements; // Dynamically allocated array of items
    int Count;      // Current count of items in the cart
} Map;
```

setmap.h

Dalam program Purrmart, kami menggunakan ADT Setmap untuk mengelola keranjang belanja pelanggan. ADT ini memanfaatkan `boolean.h` sebagai dasar untuk operasi logika dan `barang.h` untuk mendefinisikan tipe data `Barang` yang digunakan sebagai kunci dalam map. ADT ini dibuat untuk menyimpan barang yang dipilih pelanggan bersama jumlah kuantitasnya, serta mendukung operasi seperti menambah barang ke keranjang, memperbarui jumlah barang, menghapus barang tertentu, dan menghitung total harga barang dalam keranjang. Selain itu, ADT ini berfungsi untuk memastikan keranjang belanja tetap efisien dengan memanfaatkan struktur array dinamis, sehingga dapat menyesuaikan kapasitas penyimpanan sesuai kebutuhan.

ADT Setmap ini berada pada file header `setmap.h`. Di dalamnya, terdapat struktur data `Map`, yang memiliki atribut *Elements* berupa array dinamis untuk menyimpan barang dan kuantitasnya, serta atribut *Count* untuk mencatat jumlah elemen yang ada di keranjang. Setiap elemen dalam array direpresentasikan sebagai struktur `item`, yang terdiri dari *Key* bertipe `Barang` dan *Quantity* untuk jumlah barang tersebut.

3.10 ADT Linked List

SKETSA STRUKTUR DATA :

```
#define NIL NULL

typedef struct tElmtlist *addr_listdp;
typedef struct tElmtlist
{
    Barang wish;
    addr_listdp next;
    addr_listdp prev;
} ElmtList;
```

```

typedef struct
{
    addr_listdp First;
    addr_listdp Last;
    int count;
} Linkedlist;

/* Notasi Akses */
#define Wish(P) (P)->wish
#define Next(P) (P)->next
#define Prev(P) (P)->prev
#define First(L) ((L).First)
#define Last(L) ((L).Last)
#define Count(L) ((L).count)

```

linkedlist.h

Dalam program Purrmart, kami menggunakan ADT Linked List untuk mengelola data wishlist pelanggan secara dinamis. ADT ini memanfaatkan `boolean.h` sebagai dasar untuk operasi logika dan `barang.h` untuk mendefinisikan tipe data Barang yang digunakan dalam setiap elemen Linked List. ADT ini dibuat untuk mempermudah pengelolaan daftar barang yang diinginkan pelanggan, termasuk menambahkan barang baru, menghapus barang tertentu, serta mencetak seluruh isi wishlist ke layar. Selain itu, ADT ini berfungsi untuk menyimpan data secara terurut dengan referensi elemen berikutnya (*next*) dan sebelumnya (*prev*), yang mempermudah manipulasi data tanpa perlu mengalokasikan ulang seluruh struktur.

ADT Linkedlist ini berada pada file header `linkedlist.h`. Di dalamnya, terdapat struktur data Linked List, yang memiliki atribut *First* untuk menunjuk elemen pertama, *Last* untuk menunjuk elemen terakhir, serta *count* untuk mencatat jumlah elemen dalam daftar. Elemen daftar direpresentasikan sebagai struktur `ElmtList`, yang memiliki atribut *wish* bertipe Barang, serta pointer *next* dan *prev* untuk menghubungkan elemen-elemen dalam Linked List.

4 Program Utama

Program utama dimulai dengan pendefinisian header yang diperlukan untuk mendukung spesifikasi program. Header utama mencakup `<stdio.h>`, `<stdlib.h>`, dan **"boolean.h"** yang menyediakan fungsi dasar seperti input/output, alokasi memori, dan tipe data boolean. Selain itu, terdapat banyak header tambahan yang berasal dari Abstract Data Types (ADT) dan spesifikasi fungsi. Header ADT meliputi **barang.h**, **user.h**, **mesinkarakter.h**, **mesinkata.h**, **queue.h**, **arraydin.h**, **list.h**, dan **mesinkalimat.h**, sedangkan header spesifikasi mencakup berbagai fitur seperti **load.h**, **save.h**, **login.h**, **logout.h**, **help.h**, **StoreList.h**, hingga **profile.h**. Header-header ini memberikan modularitas pada kode dengan membagi fungsi ke dalam modul yang spesifik.

Fungsi utama (**main()**) dimulai dengan menampilkan antarmuka awal menggunakan fungsi **printPurrmart()**. Selanjutnya, program menginisialisasi beberapa struktur data seperti **List userList** untuk menyimpan data pengguna, **ArrayDin**

barangList untuk menyimpan daftar barang secara dinamis, serta **Queue barangQueue** untuk mengelola antrean barang. Variabel penting lainnya seperti **currentIndex** untuk melacak pengguna yang sedang aktif, **filename** untuk mencatat nama file yang digunakan, dan **level** untuk menunjukkan tingkatan menu juga diinisialisasi.

Bagian utama dari program berjalan dalam sebuah loop yang terus aktif hingga pengguna memutuskan untuk keluar. Pada Level 1, pengguna diberikan opsi untuk memulai sesi baru dengan **START**, memuat file data khusus dengan **LOAD**, meminta bantuan dengan **HELP**, atau keluar dengan **QUIT**. Jika pengguna memilih **START**, file default akan dimuat ke dalam program, sedangkan **LOAD** memungkinkan pengguna untuk menentukan file tertentu dari direktori "DATA/". Apabila perintah tidak valid, pengguna diarahkan untuk menggunakan **HELP**.

Setelah data berhasil dimuat, program beralih ke Level 2 untuk proses autentikasi pengguna. Pada tahap ini, pengguna dapat login menggunakan **LOGIN** atau mendaftar dengan **REGISTER**. Opsi bantuan tetap tersedia melalui **HELP**, dan pengguna dapat keluar kapan saja dengan **QUIT**. Apabila login berhasil, pengguna akan diarahkan ke Level 3.

Di Level 3, pengguna memiliki akses ke fitur utama program seperti **WORK**, **STORE**, **CART**, **WISHLIST**, **HISTORY**, **PROFILE**, **SAVE**, **LOGOUT**, dan **QUIT**. Fitur **WORK** memungkinkan pengguna untuk bekerja dengan opsi reguler atau tantangan. **STORE** memberikan berbagai operasi pengelolaan barang, termasuk menampilkan daftar barang, meminta barang tertentu, memasok barang ke inventaris, atau menghapus barang. Fitur **CART** dan **WISHLIST** memungkinkan pengguna untuk mengelola keranjang belanja serta daftar keinginan mereka. Selain itu, **HISTORY** menampilkan riwayat aktivitas pengguna, dan **PROFILE** menyediakan informasi lengkap tentang profil pengguna. Data yang telah diubah dapat disimpan menggunakan **SAVE**, sedangkan **LOGOUT** memungkinkan pengguna untuk keluar dari sesi mereka tanpa keluar dari program.

Setiap perintah yang valid akan memanggil fungsi atau prosedur yang relevan untuk dieksekusi. Apabila pengguna memasukkan perintah yang tidak dikenal, program akan memberikan peringatan dan meminta perintah yang valid. Sebelum keluar dari program menggunakan **QUIT**, pengguna diberikan opsi untuk menyimpan sesi mereka. Jika memilih keluar tanpa menyimpan, data sementara akan dihapus. Program berakhir dengan memanggil fungsi **printClosing()** untuk menampilkan antarmuka penutup, memastikan pengalaman pengguna yang terorganisir dan terstruktur.

Berikut ini algoritma dari program utama yang kami buat dengan file yang diberi nama "main.c" :

```
#include <stdio.h>
#include <stdlib.h>
#include "boolean.h"

/*-----HEADER ADT-----*/
#include "ADT/Kustom/barang.h"
#include "ADT/Kustom/user.h"
#include "ADT/MesinKarakter/mesinkarakter.h"
#include "ADT/MesinKata/mesinkata.h"
#include "ADT/Queue/queue.h"
```

```

#include "ADT/ListDinamis/arraydin.h"
#include "ADT/List/list.h"
#include "ADT/MesinKalimat/mesinkalimat.h"

/*-----HEADER SPESIFIKASI-----*/
#include "SPESIFIKASI/Load/load.h"
#include "SPESIFIKASI/Save/save.h"
#include "SPESIFIKASI/Login/login.h"
#include "SPESIFIKASI/Logout/logout.h"
#include "SPESIFIKASI/Help/help.h"
#include "SPESIFIKASI/Register/register.h"
#include "SPESIFIKASI/Help/help.h"
#include "SPESIFIKASI/StoreList/StoreList.h"
#include "SPESIFIKASI/StoreRemove/StoreRemove.h"
#include "SPESIFIKASI/StoreRequest/StoreRequest.h"
#include "SPESIFIKASI/StoreSupply/StoreSupply.h"
#include "SPESIFIKASI/Work/work.h"
#include "SPESIFIKASI/workchallenge/workchallenge.h"
#include "SPESIFIKASI/StoreRequestBioWeapon/StoreRequestBioWeapon.h"
#include "SPESIFIKASI/CartAdd/cartadd.h"
#include "SPESIFIKASI/CartPay/cartpay.h"
#include "SPESIFIKASI/CartRemove/cartremove.h"
#include "SPESIFIKASI/History/history.h"
#include "SPESIFIKASI/WishlistAdd/wishlistadd.h"
#include "SPESIFIKASI/WishlistClear/wishlistclear.h"
#include "SPESIFIKASI/WishlistRemove/WishlistRemove.h"
#include "SPESIFIKASI/WishlistRemoveIdx/WishlistRemoveIdx.h"
#include "SPESIFIKASI/WishlistShow/wishlistshow.h"
#include "SPESIFIKASI/WishlistSwap/WishlistSwap.h"
#include "SPESIFIKASI/profile/profile.h"

int main()
{
    printPurrmart();

    // INISIALISASI STRUKTUR DATA
    IdxType currentIndex = IDX_UNDEF;
    char filename[MAX_LEN];
    List userList = MakeList();
    ArrayDin barangList = MakeArrayDin();
    Queue barangQueue;
    CreateQueue(&barangQueue);

    // ALGORITMA

    int level = 1;
    boolean isRunning = true;

    while (isRunning)
    {
        if (level == 1)
        {
            PrintLevel1Menu();
            printf("\n> ");
            STARTWORD();

            // Tingkatan 1: START, LOAD, HELP, QUIT
            if (IsWordEqual(currentWord, StringToWord("START")))
            {
                char defaultFilename[] = "default.txt";
                char baseDir[] = "DATA/"; // Direktori default tempat file berada
                char fullPath[200];
                int i = 0, j = 0;
            }
        }
    }
}

```

```

        // Gabungkan baseDir dengan defaultFilename
        while (baseDir[i] != '\0') { // Salin baseDir ke fullPath
            fullPath[i] = baseDir[i];
            i++;
        }
        while (defaultFilename[j] != '\0') { // Salin defaultFilename ke fullPath
            fullPath[i] = defaultFilename[j];
            i++;
            j++;
        }
        fullPath[i] = '\0'; // Null-terminate string

        // Debugging: Tampilkan isi fullPath
        printf("DEBUG: Full Path = %s\n", fullPath);
        Load(fullPath, &barangList, &userList);
        level = 2;
    }
    else if (IsWordEqual(currentWord, StringToWord("LOAD")))
    {
        printf("Nama File (.txt): ");
        printf("\n> ");
        STARTWORD();
        for (int i = 0; i < currentWord.Length; i++)
        {
            filename[i] = currentWord.TabWord[i];
        }
        filename[currentWord.Length] = '\0';

        char baseDir[] = "DATA/"; // Direktori default tempat file berada
        char fullPath[200];
        int i = 0, j = 0;

        // Gabungkan baseDir dengan filename
        while (baseDir[i] != '\0') {
            fullPath[i] = baseDir[i];
            i++;
        }
        while (filename[j] != '\0') {
            fullPath[i] = filename[j];
            i++;
            j++;
        }
        fullPath[i] = '\0'; // Null-terminate string
        printf("%s", fullPath);

        Load(fullPath, &barangList, &userList);
        if (!EndKalimat)
        {
            printf("File berhasil dimuat. Masuk ke autentikasi pengguna.\n");
            level = 2;
        }
        else
        {
            printf("Gagal memuat file.\n");
        }
    }
    else if (IsWordEqual(currentWord, StringToWord("HELP")))
    {
        DisplayHelp1();
    }
    else if (IsWordEqual(currentWord, StringToWord("QUIT")))
    {
        isRunning = false;
    }

```

```

    }
    else
    {
        printf("Command tidak dikenal. Silakan masukkan command yang valid.\n");
    }
}
else if (level == 2)
{
    PrintLevel2Menu();
    printf("\n> ");
    STARTWORD();

    // Tingkatan 2: LOGIN, REGISTER, HELP, QUIT
    if (IsWordEqual(currentWord, StringToWord("LOGIN")))
    {
        Login(&userList, &currentIndex);
        ;
        level = 3;
    }
    else if (IsWordEqual(currentWord, StringToWord("REGISTER")))
    {
        RegisterUser(&userList);
    }
    else if (IsWordEqual(currentWord, StringToWord("HELP")))
    {
        DisplayHelp2();
    }
    else if (IsWordEqual(currentWord, StringToWord("QUIT")))
    {
        isRunning = false;
    }
    else
    {
        printf("Command tidak dikenal. Silakan masukkan command yang valid.\n");
    }
}
else if (level == 3)
{
    PrintLevel3Menu();
    printf("\n> ");
    STARTWORD();

    // Tingkatan 3: WORK, STORE, CART, WISHLIST, LOGOUT, SAVE, QUIT
    if (IsWordEqual(currentWord, StringToWord("WORK")))
    {
        ADVWORD();
        if (IsWordEqual(currentWord, StringToWord("CHALLENGE")))
        {
            WorkChallenge(&userList, currentIndex); // Done testing
        }
        else
        {
            Work(&userList, currentIndex);
        }
    }
    else if (IsWordEqual(currentWord, StringToWord("STORE")))
    {
        ADVWORD();
        if (IsWordEqual(currentWord, StringToWord("LIST")))
        {
            StoreList(&barangList);
        }
        else if (IsWordEqual(currentWord, StringToWord("REQUEST")))
        {

```



```

        ADVWORD();
        if (IsWordEqual(currentWord, StringToWord("BIOWEAPON"))){
            StoreRequestBioWeapon(&barangQueue, &barangList);
        }
        else if (endWord)
        {
            StoreRequest(&barangQueue, &barangList);
        }
        else{
            printf("Command tidak dikenal.\n");
        }
    }
else if (IsWordEqual(currentWord, StringToWord("SUPPLY"))){
    StoreSupply(&barangQueue, &barangList);
}
else if (IsWordEqual(currentWord, StringToWord("REMOVE"))){
    StoreRemove(&barangList);
}
else
{
    printf("Command tidak dikenal.\n");
}
}
else if (IsWordEqual(currentWord, StringToWord("CART"))){
{
    ADVWORD();
    printf("DEBUG: Inside CART block. CurrentWord: ");
    PrintWord(currentWord);
    printf("\n");
    if (IsWordEqual(currentWord, StringToWord("ADD"))){
        printf("DEBUG: CART ADD detected.\n");
        ADVWORD();
        CartAdd(&userList, currentIndex, barangList);
    }
    else if (IsWordEqual(currentWord, StringToWord("PAY"))){
        CartPay(&userList, currentIndex);
    }
    else if (IsWordEqual(currentWord, StringToWord("REMOVE"))){
        CartRemove(&userList, currentIndex);
    }
    else if (IsWordEqual(currentWord, StringToWord("SHOW"))){
        TampilkanKeranjang(userList.A[currentIndex].keranjang);
    }
    else
    {
        printf("Command tidak dikenal.\n");
    }
}
}
else if (IsWordEqual(currentWord, StringToWord("WISHLIST"))){
{
    ADVWORD();
    printf("DEBUG: Inside CART block. CurrentWord: ");
    PrintWord(currentWord);
    printf("\n");
    if (IsWordEqual(currentWord, StringToWord("ADD"))){
        printf("DEBUG: CART ADD detected.\n");
        ADVWORD();
    }
}
}

```

```

        WishlistAdd(&userList, currentIndex, barangList);
    }
    else if (IsWordEqual(currentWord, StringToWord("SHOW")))
    {
        wishlistShow(&userList.A[currentIndex].wishlist);
    }
    else if (IsWordEqual(currentWord, StringToWord("CLEAR")))
    {
        WishlistClear(&(userList.A[currentIndex].wishlist));
    }
    else if (IsWordEqual(currentWord, StringToWord("REMOVE")))
    {
        ADVWORD();
        if (!endWord){
            int idxToRemove = WordToInt(currentWord);
            WishlistRemoveIdx(&userList.A[currentIndex].wishlist, idxToRemove);
        }
        else{
            printf("Masukkan nama barang yang akan dihapus: ");
            STARTWORD();
            Word words[100];
            int wordCount = 0;

            // Simpan seluruh kata
            while (!endWord) {
                words[wordCount] = currentWord;
                wordCount++;
                ADVWORD();
            }

            PrintWord(*words);

            if (wordCount < 1) {
                printf("Format command tidak valid!\n");
                return 0;
            }

            // Inisialisasi nama barang sebagai Word kosong
            Word namaBarang;
            ResetWord(&namaBarang);

            // Gabungkan semua kata kecuali kata terakhir untuk nama barang
            for (int i = 0; i < wordCount; i++) {
                for (int j = 0; j < words[i].Length; j++) {
                    namaBarang.TabWord[namaBarang.Length++] =
words[i].TabWord[j];
                }
                if (i < wordCount - 1) {
                    namaBarang.TabWord[namaBarang.Length++] = ' ';
                }
            }
            WishlistRemove(&userList.A[currentIndex].wishlist, namaBarang);
        }
    }
    else if (IsWordEqual(currentWord, StringToWord("SWAP")))
    {
        ADVWORD();
        int i = WordToInt(currentWord);
        ADVWORD();
        int j = WordToInt(currentWord);
        WishlistSwap(&(userList.A[currentIndex].wishlist), i, j);
    }
    else
    {

```

```

        printf("Command tidak dikenal.\n");
    }
}
else if (IsWordEqual(currentWord, StringToWord("HISTORY")))
{
    ADVWORD();
    history(&userList, currentIndex);
}
else if (IsWordEqual(currentWord, StringToWord("PROFILE")))
{
    showProfile(userList, currentIndex);
}
else if (IsWordEqual(currentWord, StringToWord("LOGOUT")))
{
    Logout(currentIndex);
    level = 2;
}
else if (IsWordEqual(currentWord, StringToWord("SAVE")))
{
    Save(&barangList, &userList, filename);
    printf("Berhasil menyimpan data!");
}
else if (IsWordEqual(currentWord, StringToWord("QUIT")))
{
    printf("Apakah Anda ingin menyimpan sesi ini? (Y/N)\n");
    boolean quit = false;
    while (!quit)
    {
        printf("> ");
        STARTWORD();
        if (IsWordEqual(currentWord, StringToWord("Y")))
        {
            Save(&barangList, &userList, filename);
            printf("Berhasil menyimpan data!");
            quit = true;
        }
        else if (IsWordEqual(currentWord, StringToWord("N")))
        {
            quit = true;
        }
        else
        {
            printf("Command tidak dikenal. Silakan masukkan command yang
valid.\n");
            printf("Apakah Anda ingin menyimpan sesi ini? (Y/N)\n");
        }
        isRunning = false;
    }
}
else if (IsWordEqual(currentWord, StringToWord("HELP")))
{
    DisplayHelp3();
}
else
{
    printf("Command tidak dikenal. Silakan masukkan command yang valid.\n");
}
}

printClosing();
return 0;
}

```

5 Algoritma-Algoritma Menarik

5.1 Read Integer Name Items di Cart Remove

```
Word words[100]; //menyimpan words dalam bentuk array
int wordCount = 0;

//Simpan seluruh word
while (!endWord) {
    words[wordCount] = currentWord;
    wordCount++;
    ADVWORDNotIgnore();
}

if (wordCount < 2) {
    printf("Format command tidak valid!\n");
    return;
}

// Clear currentBarang.name
for (int i = 0; i < MAX_LEN; i++) {
    currentBarang.name[i] = '\0';
}

int idx = 0;
// Semua word ditambah kecuali word paling akhir
for (int i = 0; i < wordCount - 1; i++) {
    // Append current word ke currentBarang.name
    for (int j = 0; j < words[i].Length && idx < MAX_LEN - 1; j++) {
        currentBarang.name[idx++] = words[i].TabWord[j];
    }

    // spasi kalau bukan kata terakhir
    if (i < wordCount - 2 && idx < MAX_LEN - 1) {
        currentBarang.name[idx++] = ' ';
    }
}

currentBarang.name[idx] = '\0';
```

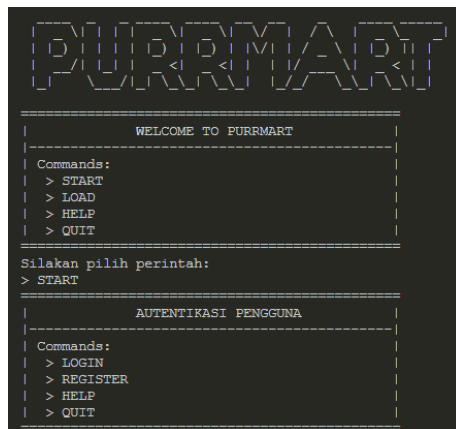
Algoritma ini dirancang untuk memproses input berupa kumpulan kata, termasuk **angka**, dan menggabungkannya menjadi sebuah **nama barang**. Setiap kata dibaca satu per satu menggunakan fungsi **ADVWORDNotIgnore()** dan disimpan ke dalam array **words**. Kemudian, algoritma memastikan bahwa input memiliki lebih dari satu kata; jika tidak, proses dihentikan dengan pesan **kesalahan**. Setelah itu, **nama barang** yang lama dibersihkan dengan mengganti semua elemen array-nya menjadi karakter **null** ('\0'). Kata-kata dalam array digabungkan menjadi **string** baru dengan menambahkan setiap **karakter** dari kata, termasuk **angka**, ke dalam **currentBarang.name**. **Spasi** ditambahkan di antara kata-kata kecuali sebelum kata terakhir. Proses ini diawasi agar

panjang **nama barang** tidak melebihi batas maksimum (**MAX_LEN**). Algoritma ini **fleksibel** karena memperlakukan **angka** dalam kata sebagai bagian dari **string**, sehingga cocok untuk menangani **nama barang** yang sering kali mengandung kombinasi **huruf** dan **angka**, seperti "iPhone 14" atau "PS5".

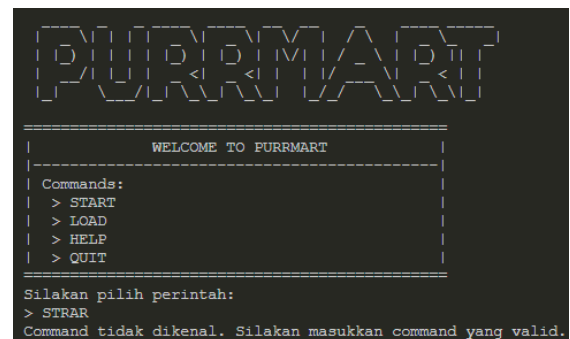
6 Data Test

6.1 START

Program Purrmart dimulai dengan memasukkan command **START** pada terminal yang telah di-*run* sebelumnya. Perlu diperhatikan bahwa penulisan command harus sesuai dengan opsi yang ada, jika tidak maka program akan meminta input ulang seperti pada *Gambar 6.1.1*. command ini digunakan untuk membaca file konfigurasi default yang tersimpan dalam **default.txt**. File ini berisi daftar barang-barang yang tersedia di toko dan dapat digunakan selama permainan berlangsung. Setelah file konfigurasi berhasil dibuka, program akan menampilkan pesan bahwa file telah berhasil dibaca, diikuti dengan daftar command yang bisa digunakan. Daftar command yang tersedia yaitu **LOGIN** untuk masuk ke akun, **REGISTER** untuk mendaftarkan akun baru, **HELP** untuk melihat panduan penggunaan, dan **QUIT** untuk keluar dari program.



• *Gambar 6.1.1 Menampilkan START yang valid*



Gambar 6.1.2 Menampilkan START yang tidak valid

6.2 LOAD

Command **LOAD** pada opsi Purrmart digunakan ketika pengguna ingin memulai program dengan file konfigurasi buatan sendiri, bukan file konfigurasi default. Dengan menggunakan command ini, pengguna dapat menentukan daftar barang yang akan tersedia selama permainan sesuai pilihan pengguna. Untuk menjalankan command ini, pengguna harus memasukkan nama file konfigurasi yang diinginkan dengan format **.txt**, dan apabila tidak sesuai maka pengguna harus melakukan input file kembali seperti pada *Gambar 6.2.2*. Setelah file tersebut dimasukkan, program akan memeriksa apakah file tersebut

dapat dibaca. Jika file berhasil dimuat, program akan menampilkan pesan: *"File berhasil dimuat. Masuk ke autentikasi pengguna."*

```
Silakan pilih perintah:
> LOAD
Nama File (.txt):
> default.txt
DATA/default.txt File berhasil dimuat. Masuk ke autentikasi pengguna.
```

```
Silakan pilih perintah:
> LOAD
Nama File (.txt):
> default.pdf
DATA/default.pdfERROR: File tidak valid atau tidak ditemukan.
Gagal memuat file.
```

- *Gambar 6.2.1 Menampilkan LOAD dengan file konfigurasi yang valid*
- *Gambar 6.2.2 Menampilkan LOAD dengan file konfigurasi yang tidak valid*

6.3 SAVE

Setelah pengguna berhasil melakukan **LOGIN**, command **SAVE** dapat digunakan untuk menyimpan seluruh data yang sudah ter-*update* selama menggunakan fitur di dalam Purmart kedalam file **.txt** yang baru sehingga nama file konfigurasi harus bersifat unik dan belum pernah digunakan seperti pada *Gambar 6.3.1*, maka akan menampilkan *"Berhasil menyimpan data!"*. Namun, apabila file konfigurasi telah digunakan atau tidak dalam format **.txt** seperti pada *Gambar 6.3.2*, maka akan menampilkan *"Nama file harus berakhiran .txt!"* dan *"Nama file tidak boleh berawalan '<Nama file>'!"*

```
Silakan pilih perintah:
> SAVE
Masukkan nama file untuk menyimpan data (harus .txt dan tidak berawalan 'default')
Save file berhasil disimpan pada dummy.txt
Berhasil menyimpan data!
```

```
Silakan pilih perintah:
> SAVE
Masukkan nama file untuk menyimpan data (harus .txt dan tidak berawalan 'da
Nama file harus berakhiran .txt!
Silakan coba lagi.
```

- *Gambar 6.3.1 Menampilkan SAVE yang valid*
- *Gambar 6.3.2 Menampilkan SAVE yang tidak valid*

6.4 STORE LIST

Setelah pengguna berhasil melakukan **LOGIN**, command **STORE LIST** dapat digunakan untuk menampilkan daftar barang yang tersedia di toko dan setiap barang bersifat unik sehingga tidak terdapat duplikasi seperti pada *Gambar 7.4.1*.

```
Silakan pilih perintah:
> STORE LIST
List barang yang ada di toko:
+-----+-----+
| Nama Barang | Harga |
+-----+-----+
| AK47        | 10    |
| Lalabu      | 20    |
| Ayam Goreng Crisbar | 30    |
| Meong       | 50    |
+-----+-----+
```

- *Gambar 6.4.1 Menampilkan STORE LIST*

6.5 PROFILE

Setelah pengguna berhasil melakukan **LOGIN**, command **PROFILE** dapat digunakan untuk menampilkan informasi akun pengguna. Sistem akan mencetak data lengkap pengguna, termasuk nama pengguna, saldo yang tersedia, isi wishlist, dan isi keranjang belanja. Jika wishlist kosong, sistem akan menampilkan pesan *"Wishlist kamu"*

kosong!” seperti pada *Gambar 6.5.1*. Begitu pula jika keranjang belanja kosong, sistem akan menampilkan pesan “*Keranjang kamu kosong!*” seperti pada *Gambar 6.5.2*. Namun, jika terdapat barang di wishlist atau keranjang, sistem akan mencetak daftar barang tersebut dalam format yang rapi. Fitur ini membantu pengguna untuk melihat informasi akun mereka secara keseluruhan dalam satu perintah, memberikan kemudahan untuk memantau status akun tanpa perlu menggunakan beberapa command terpisah.

```
Silakan pilih perintah:
> PROFILE
Nama      : admin
Password  : alstrukdatkeren
Saldo     : 100

Isi Keranjang:
+-----+-----+-----+
| Kuantitas | Nama Barang | Total |
+-----+-----+-----+
| 2         | Lalabu     | 40    |
| 6         | AK47       | 60    |
+-----+-----+-----+

Wishlist:
Wishlist kosong.
```

● *Gambar 6.5.1 Menampilkan PROFILE dengan Wishlist yang kosong*

```
Silakan pilih perintah:
> PROFILE
Nama      : admin
Password  : alstrukdatkeren
Saldo     : 100

Isi Keranjang:
Keranjang kosong.

Wishlist:
1. Meong
2. Lalabu
```

● *Gambar 6.5.1 Menampilkan PROFILE dengan keranjang kosong*

6.6 CART ADD <nama> <n>

Setelah pengguna berhasil melakukan **LOGIN**, command **CART ADD** dapat digunakan untuk menambahkan barang ke dalam keranjang belanja. Pengguna perlu memasukkan nama barang dan jumlah yang ingin ditambahkan ke keranjang dalam format **CART ADD <nama_barang> <jumlah>**. Sistem akan memvalidasi keberadaan barang di toko berdasarkan input pengguna. Jika barang ditemukan, sistem akan menambahkan barang ke keranjang dengan jumlah yang sesuai seperti pada *Gambar 6.6.1*, dan akan menampilkan pesan “*Berhasil menambahkan <jumlah> <nama_barang> ke keranjang belanja!*”. Namun, jika barang yang dimasukkan tidak ada dalam daftar toko, sistem akan menampilkan pesan kesalahan “*Barang tidak ditemukan di toko!*” seperti pada *Gambar 6.6.2*.

```
Silakan pilih perintah:
> CART ADD AK47 5
Keranjang count: 1
Barang: AK47, Harga: 10, Jumlah: 5
Berhasil menambahkan 5 AK47 ke keranjang belanja!
```

● *Gambar 6.6.1 Menampilkan CART ADD yang valid*

```
Silakan pilih perintah:
> CART ADD Ambalabu 10
[ERROR] Barang 'Ambalabu' tidak ditemukan di toko.
```

Gambar 6.6.2 Menampilkan CART ADD yang tidak valid

6.7 CART REMOVE <nama> <n>

Setelah pengguna berhasil melakukan **LOGIN**, command **CART REMOVE <nama_barang> <jumlah>** dapat digunakan untuk menghapus barang tertentu dari keranjang belanja. Sistem akan memeriksa keberadaan barang yang dimaksud di dalam

keranjang. Jika barang tidak ada di keranjang, sistem akan menampilkan pesan “*Barang tidak ada di keranjang belanja!*” seperti pada *Gambar 6.7.1*. Jika barang ditemukan di keranjang, sistem akan memeriksa kuantitas barang. Jika jumlah yang diminta untuk dihapus kurang dari atau sama dengan jumlah barang yang ada di keranjang, sistem akan mengurangi kuantitas barang tersebut sesuai input. Jika jumlah yang diminta sama dengan jumlah barang di keranjang, barang akan dihapus sepenuhnya dari keranjang, dan sistem akan menampilkan pesan “*Berhasil mengurangi <jumlah> <nama_barang> dari keranjang belanja!*” seperti pada *Gambar 6.7.2*. Namun, jika jumlah yang diminta untuk dihapus lebih besar daripada jumlah barang yang tersedia, sistem akan menampilkan pesan “*Tidak berhasil mengurangi, hanya terdapat <jumlah_barang_di_keranjang> <nama_barang> pada keranjang!*” untuk memberi tahu pengguna tentang kesalahan input seperti pada *Gambar 6.7.3*.

```
Silakan pilih perintah:
> CART REMOVE Koming 2
Barang tidak ada di keranjang belanja!
```

- *Gambar 6.7.1 Menampilkan CART REMOVE dengan barang tidak berada di keranjang*

```
Silakan pilih perintah:
> CART REMOVE AK47 3
Berhasil mengurangi 3 AK47 dari keranjang belanja!
```

- *Gambar 6.7.2 Menampilkan CART REMOVE yang valid*

```
Silakan pilih perintah:
> CART REMOVE AK47 12
Tidak berhasil mengurangi, hanya terdapat 5 AK47 pada keranjang!
```

- *Gambar 6.7.3 Menampilkan CART REMOVE yang*
- *dengan jumlah barang kurang dari jumlah di keranjang*

6.8 CART SHOW

Setelah pengguna berhasil melakukan **LOGIN**, command **CART SHOW** dapat digunakan untuk melihat daftar barang yang ada di keranjang belanja pengguna. Sistem akan memeriksa apakah keranjang belanja pengguna kosong. Jika keranjang kosong, sistem akan menampilkan pesan “*Keranjang kamu kosong!*” seperti pada *Gambar 6.8.1*. Namun, jika terdapat barang di keranjang, sistem akan mencetak daftar barang dalam format tabel yang berisi nama barang, kuantitas, dan subtotal harga untuk setiap barang seperti pada *Gambar 6.8.2*. Selain itu, sistem akan menampilkan total harga keseluruhan barang yang ada di keranjang, sehingga pengguna dapat dengan mudah mengetahui jumlah biaya yang diperlukan jika ingin melakukan pembayaran.

```
Silakan pilih perintah:
> CART SHOW
Keranjang kosong.
```

- *Gambar 6.8.1 Menampilkan CART SHOW apabila keranjang kosong*

```
Silakan pilih perintah:
> CART SHOW
+-----+-----+-----+
| Kuantitas | Nama Barang | Total |
+-----+-----+-----+
| 10        | Lalabu     | 200   |
| 2         | AK47       | 20    |
+-----+-----+-----+
```

- *Gambar 6.8.2 Menampilkan CART SHOW yang valid*

6.9 CART PAY

Setelah pengguna berhasil melakukan **LOGIN**, command **CART PAY** dapat digunakan untuk menyelesaikan pembelian seluruh barang yang ada di keranjang belanja. Pengguna akan diminta konfirmasi dengan pilihan “Ya” atau “Tidak” untuk melanjutkan pembayaran. Jika pengguna memilih “Ya” dan saldo mencukupi, sistem akan mengurangi saldo pengguna sebesar total harga keranjang, mengosongkan keranjang, dan menampilkan pesan “*Selamat, kamu telah membeli barang-barang tersebut!*” seperti pada *Gambar 6.9.1*. Selain itu, barang dengan total harga terbesar akan ditambahkan ke dalam riwayat pembelian pengguna untuk memberikan informasi prioritas pembelian. Jika saldo pengguna tidak mencukupi, sistem akan menampilkan pesan kesalahan “*Uang kamu hanya <saldo>, tidak cukup untuk membeli keranjang!*” seperti pada *Gambar 6.9.2*. Jika pengguna memilih “Tidak” atau memberikan input yang tidak valid, sistem akan membatalkan pembelian dan menampilkan pesan “*Pembelian dibatalkan.*” seperti pada *Gambar 2.9.3*.

```
Silakan pilih perintah:
> CART PAY
Kamu akan membeli barang-barang berikut:
+-----+-----+-----+
| Kuantitas | Nama Barang | Total |
+-----+-----+-----+
| 2         | AK47       | 20    |
+-----+-----+-----+
Total biaya yang harus dikeluarkan adalah 20, apakah jadi dibeli? (Ya/Tidak): Ya
Selamat, kamu telah membeli barang-barang tersebut!
```

● *Gambar 6.9.1 Menampilkan CART PAY apabila saldo cukup*

```
Silakan pilih perintah:
> CART PAY
Kamu akan membeli barang-barang berikut:
+-----+-----+-----+
| Kuantitas | Nama Barang | Total |
+-----+-----+-----+
| 10        | Lalabu     | 200   |
| 2         | AK47       | 20    |
+-----+-----+-----+
Total biaya yang harus dikeluarkan adalah 220, apakah jadi dibeli? (Ya/Tidak): Ya
Uang kamu hanya 100, tidak cukup untuk membeli keranjang!
```

Gambar 6.9.2 Menampilkan CART PAY apabila saldo tidak cukup

```
Silakan pilih perintah:
> CART PAY
Kamu akan membeli barang-barang berikut:
+-----+-----+-----+
| Kuantitas | Nama Barang | Total |
+-----+-----+-----+
| 5         | Lalabu     | 100   |
+-----+-----+-----+
Total biaya yang harus dikeluarkan adalah 100, apakah jadi dibeli? (Ya/Tidak): Tidak
Pembelian dibatalkan.
```

● *Gambar 6.9.2 Menampilkan CART PAY apabila dibatalkan*

6.10 HISTORY <n>

Setelah pengguna berhasil melakukan **LOGIN**, command **HISTORY <jumlah>** dapat digunakan untuk menampilkan riwayat pembelian barang yang telah dilakukan oleh pengguna. Sistem akan memeriksa apakah pengguna memiliki riwayat pembelian. Jika tidak ada barang yang pernah dibeli, sistem akan menampilkan pesan seperti pada *Gambar 6.10.1*. Jika pengguna memiliki riwayat pembelian, sistem akan mencetak daftar barang yang telah dibeli, dimulai dari pembelian paling baru hingga yang lebih lama, dengan jumlah yang ditampilkan sesuai input <jumlah>. Setiap entri dalam riwayat akan memuat nama barang dan total harga, ditampilkan dalam format tabel seperti pada *Gambar 6.10.2*.

```
Silakan pilih perintah:
> HISTORY
Riwayat pembelian barang:
+-----+-----+
| Nama Barang | Total Harga |
+-----+-----+
|              |              |
+-----+-----+
```

Gambar 6.10.1 Menampilkan CART HISTORY apabila belum melakukan pembelian

```
Silakan pilih perintah:
> HISTORY
Riwayat pembelian barang:
+-----+-----+
| Nama Barang | Total Harga |
+-----+-----+
| AK47        | 10           |
| Ayam Goreng Crisbar | 30           |
| Lalabu      | 20           |
| AK47        | 10           |
+-----+-----+
```

Gambar 6.10.2 Menampilkan CART HISTORY apabila telah melakukan pembelian

6.11 WISHLIST ADD

Setelah pengguna berhasil melakukan **LOGIN**, command **WISHLIST ADD** dapat digunakan untuk menambahkan barang wishlist. Pengguna perlu memasukkan nama barang dan jumlah yang ingin ditambahkan ke keranjang dalam format **CART ADD <nama_barang> <jumlah>**. Sistem akan memvalidasi keberadaan barang di toko berdasarkan input pengguna. Jika barang ditemukan, sistem akan menambahkan barang ke keranjang dengan jumlah yang sesuai seperti pada Gambar 6.11.1, dan akan menampilkan pesan “Berhasil menambahkan <jumlah> <nama_barang> ke keranjang belanja!”. Namun, jika barang yang dimasukkan tidak ada dalam daftar toko, sistem akan menampilkan pesan kesalahan “Tidak ada barang dengan nama <nama_barang> di toko!” seperti pada Gambar 6.11.2. Selain itu, jika barang sudah ada di wishlist, sistem akan menampilkan pesan “<nama_barang> sudah ada di wishlist!” seperti pada Gambar 6.11.3

```
Silakan pilih perintah:
> WISHLIST ADD
Masukkan nama barang: AK47
Berhasil menambahkan AK47 ke wishlist!
```

Gambar 6.11.1 Menampilkan WISHLIST ADD dengan barang yang valid

```
Silakan pilih perintah:
> WISHLIST ADD
Masukkan nama barang: Ambalabu
Tidak ada barang dengan nama 'Ambalabu' di toko!
```

Gambar 6.11.2 Menampilkan WISHLIST ADD dengan barang tidak ada di toko

```
Silakan pilih perintah:
> WISHLIST ADD
Masukkan nama barang: Lalabu
Lalabu sudah ada di wishlist!
```

Gambar 6.11.3 Menampilkan WISHLIST ADD dengan barang sudah ada di wishlist

6.12 WISHLIST SWAP <i> <j>

Setelah pengguna berhasil melakukan **LOGIN**, command **WISHLIST SWAP <i> <j>** dapat digunakan untuk menukar posisi dua barang dalam wishlist pengguna. Sistem akan memeriksa apakah wishlist pengguna memiliki isi. Jika wishlist kosong, sistem akan menampilkan pesan “Gagal menukar posisi barang, wishlist kosong!” seperti pada Gambar 6.12.1. Jika wishlist hanya memiliki satu barang, sistem akan menampilkan pesan “Gagal menukar posisi <nama_barang>! Hanya terdapat satu

barang pada wishlist sehingga posisinya tidak dapat ditukar.”. Selain itu, jika pengguna memasukkan indeks i atau j yang tidak valid, seperti kurang dari 1 atau lebih besar dari jumlah barang di wishlist, sistem akan menampilkan pesan “Gagal menukar barang, karena posisi tidak valid!” seperti pada Gambar 6.12.2. Namun, jika kedua indeks valid dan berbeda, sistem akan menukar posisi dua barang sesuai indeks yang dimasukkan, dan menampilkan pesan “Berhasil menukar posisi <nama_barang_i> dengan <nama_barang_j> pada wishlist!” seperti pada Gambar 6.12.3.

```
Silakan pilih perintah:
> WISHLIST SWAP 1 3
Gagal menukar posisi barang, wishlist kosong!
```

• Gambar 6.12.1 Menampilkan WISHLIST SWAP dengan wishlist kosong

```
Silakan pilih perintah:
> WISHLIST SWAP 1 10
Gagal menukar barang, karena posisi tidak valid!
```

• Gambar 6.12.2 Menampilkan WISHLIST SWAP dengan urutan barang tidak valid

```
Silakan pilih perintah:
> WISHLIST SWAP 1 2
Berhasil menukar posisi Meong dengan Lalabu pada wishlist!
```

• Gambar 6.12.3 Menampilkan WISHLIST SWAP yang valid

6.13 WISHLIST REMOVE

Setelah pengguna berhasil melakukan LOGIN, command WISHLIST REMOVE <nama_barang> dapat digunakan untuk menghapus barang tertentu dari wishlist pengguna. Sistem akan memeriksa apakah wishlist pengguna kosong. Jika wishlist kosong, sistem akan menampilkan pesan “Wishlist kosong! Tidak ada barang yang dapat dihapus.” seperti pada Gambar 6.13.1. Namun, jika wishlist memiliki isi, sistem akan mencari barang berdasarkan nama yang dimasukkan oleh pengguna. Jika barang ditemukan, sistem akan menghapus barang tersebut dari wishlist dan menampilkan pesan “<Nama_barang> berhasil dihapus dari WISHLIST!” seperti pada Gambar 6.13.2. Sebaliknya, jika barang yang dicari tidak ditemukan di wishlist, sistem akan menampilkan pesan “Penghapusan barang WISHLIST gagal dilakukan, <Nama_barang> tidak ada di WISHLIST!” untuk memberi tahu pengguna bahwa barang yang dimaksud tidak tersedia. Namun jika nama barang valid maka akan menampilkan pesan “<nama_barang> berhasil dihapus dari WISHLIST” seperti pada Gambar 6.13.3.

```
Silakan pilih perintah:
> WISHLIST REMOVE
Masukkan nama barang yang akan dihapus: Lalabu
Wishlist kosong! Tidak ada barang yang dapat dihapus.
```

• Gambar 6.13.1 Menampilkan WISHLIST REMOVE apabila barang pada wishlist kosong

```
Silakan pilih perintah:
> WISHLIST REMOVE
Masukkan nama barang yang akan dihapus: Ambalabu
Penghapusan barang WISHLIST gagal dilakukan, Ambalabu tidak ada di WISHLIST!
```

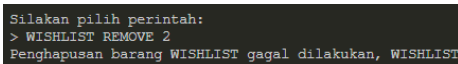
• Gambar 6.13.2 Menampilkan WISHLIST REMOVE dengan barang tidak ada pada wishlist

```
Silakan pilih perintah:
> WISHLIST REMOVE
Masukkan nama barang yang akan dihapus: Ayam Goreng Crisbar
Ayam Goreng Crisbar berhasil dihapus dari WISHLIST!
```

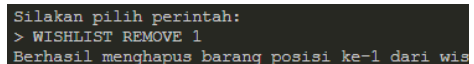
- *Gambar 6.13.3 Menampilkan WISHLIST REMOVE yang valid*

6.14 WISHLIST REMOVE <i>

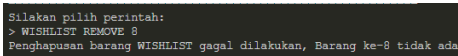
Setelah pengguna berhasil melakukan **LOGIN**, command **WISHLIST REMOVE <i>** dapat digunakan untuk menghapus barang dengan posisi tertentu dari wishlist pengguna. Sistem akan memeriksa apakah wishlist pengguna kosong. Jika wishlist kosong, sistem akan menampilkan pesan *“Wishlist kosong! Tidak ada barang yang dapat dihapus.”* seperti pada *Gambar 6.14.1*. Namun, jika wishlist memiliki isi, sistem akan mencari barang berdasarkan posisi yang dimasukkan oleh pengguna. Jika posisi barang ditemukan, sistem akan menghapus barang tersebut dari wishlist dan menampilkan pesan *“Berhasil menghapus barang posisi ke-<i> dari wishlist!”* seperti pada *Gambar 6.14.2*. Sebaliknya, jika posisi barang yang dicari tidak ditemukan di wishlist, sistem akan menampilkan pesan *“Penghapusan barang WISHLIST gagal dilakukan, Barang ke-<i> tidak ada di WISHLIST!”* untuk memberi tahu pengguna bahwa posisi barang yang dimaksud tidak tersedia seperti pada *Gambar 6.14.3*. Selanjutnya jika posisi barang yang diinput bukanlah merupakan angka maka akan menampilkan pesan *“Penghapusan barang WISHLIST gagal dilakukan, command tidak valid!”* seperti pada *Gambar 6.14.4* yang memberitahu bahwa command yang digunakan tidak valid.

- 


- *Gambar 6.14.1 Menampilkan WISHLIST REMOVE dengan wishlist kosong*

- 

- *Gambar 6.14.2 Menampilkan WISHLIST REMOVE yang valid*

- 

- *Gambar 6.14.3 Menampilkan WISHLIST REMOVE yang posisi ke-<i> tidak ditemukan*

- 

- *Gambar 6.14.4 Menampilkan WISHLIST REMOVE yang tidak memasukkan input angka*

6.15 WISHLIST CLEAR

Setelah pengguna berhasil melakukan **LOGIN**, command **WISHLIST CLEAR** dapat digunakan untuk menghapus seluruh isi wishlist pengguna. Sistem akan memeriksa apakah wishlist pengguna memiliki isi. Jika wishlist kosong, sistem akan menampilkan pesan *“Wishlist telah kosong sebelumnya!”* seperti pada *Gambar 6.15.1*. Sama hal nya apabila terdapat barang di wishlist yang kemudian dikosongkan maka sistem akan menghapus seluruh barang yang ada dan menampilkan pesan yang sama.

```
Silakan pilih perintah:
> WISHLIST CLEAR
Wishlist telah dikosongkan.
```

- *Gambar 6.15.1 Menampilkan WISHLIST CLEAR*

6.16 WISHLIST SHOW

Setelah pengguna berhasil melakukan **LOGIN**, command **WISHLIST SHOW** dapat digunakan untuk menampilkan daftar barang yang ada di wishlist pengguna. Sistem akan memeriksa apakah wishlist pengguna kosong. Jika wishlist kosong, sistem akan menampilkan pesan “*Wishlist kamu kosong!*” seperti pada *Gambar 6.16.1*. Namun, jika terdapat barang di wishlist, sistem akan mencetak daftar barang dengan format berurutan, diawali dengan nomor urut diikuti nama barang seperti pada *Gambar 6.16.2*.

```
Silakan pilih perintah:
> WISHLIST SHOW
Wishlist kamu kosong!
```

Gambar 6.16.1 Menampilkan WISHLIST SHOW apabila barang kosong

```
Silakan pilih perintah:
> WISHLIST SHOW
Berikut adalah isi wishlist-mu:
1. Meong
2. Lalabu
3. AK47
4. Ayam Goreng Crisbar
```

Gambar 6.16.2 Menampilkan WISHLIST SHOW yang valid

7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Fitur START	Memastikan aplikasi dapat dimulai dan menampilkan menu utama.	Mengetikkan command START di awal aplikasi.	START	Sistem berhasil memulai aplikasi, menampilkan menu utama, dan memungkinkan pengguna melanjutkan operasi.	Sesuai yang diharapkan.
2	Fitur LOAD	Memastikan pengguna dapat memuat file konfigurasi ke dalam sistem.	Mengetikkan command LOAD dilanjutkan dengan nama file konfigurasi yang ingin dimuat.	Load	Data dari file konfigurasi berhasil dimuat ke dalam sistem, dan output menampilkan	Sesuai yang diharapkan.

					data yang terupdate.	
3	Fitur SAVE	Memastikan pengguna dapat menyimpan seluruh data ke file konfigurasi baru.	Mengetikkan command SAVE dilanjutkan dengan nama file konfigurasi baru.	SAVE	Data berhasil disimpan ke file baru dengan format .txt, dan sistem menampilkan pesan konfirmasi.	
4	Fitur STORE LIST	Memastikan daftar barang di toko ditampilkan dengan benar.	Mengetikkan command STORE LIST.	STORE LIST	Sistem berhasil menampilkan daftar barang di toko sesuai data yang ada.	Sesuai yang diharapkan.
6	Fitur PROFILE	Memastikan pengguna dapat melihat informasi profilnya.	Mengetikkan command PROFILE.	PROFILE	Sistem menampilkan profil pengguna, termasuk nama, saldo, dan riwayat aktivitas.	Sesuai yang diharapkan.
7	Fitur CART ADD <nama> <n>	Memastikan pengguna dapat menambahkan barang ke keranjang dengan jumlah yang sesuai.	Mengetikkan command CART ADD dilanjutkan dengan nama barang dan jumlah barang yang diinginkan.	CART ADD <nama> <n>	Barang berhasil ditambahkan ke keranjang dengan jumlah yang sesuai.	Sesuai yang diharapkan.
8	Fitur CART REMOVE <nama> <n>	Memastikan pengguna dapat menghapus barang dari keranjang belanja.	Mengetikkan command CART REMOVE dilanjutkan dengan nama barang.	CART REMOVE <nama> <n>	Barang berhasil dihapus dari keranjang, dan sistem memperbarui isi keranjang.	Sesuai yang diharapkan.
9	Fitur CART SHOW	Memastikan pengguna dapat melihat isi keranjang belanja	.Mengetikkan command CART SHOW.	CART SHOW	Sistem berhasil menampilkan semua barang yang ada di keranjang, termasuk	Sesuai yang diharapkan.

		secara lengkap.			nama barang, jumlah, dan total harga.	
10	Fitur CART PAY	Memastikan pengguna dapat membayar seluruh isi keranjang jika saldo mencukupi.	Mengetikkan command CART PAY dilanjutkan dengan konfirmasi “Ya” untuk menyelesaikan pembelian.	CART PAY	Barang di keranjang berhasil dibeli, saldo berkurang sesuai total harga, dan keranjang dikosongkan.	Sesuai yang diharapkan.
11	Fitur HISTORY <n>	Memastikan pengguna dapat melihat riwayat pembelian barang.	Mengetikkan command HISTORY dilanjutkan dengan jumlah riwayat pembelian yang ingin ditampilkan.	HISTORY <n>	Riwayat pembelian barang ditampilkan sesuai jumlah yang diminta dalam format tabel.	Sesuai yang diharapkan.
12	Fitur WISHLIST ADD	Memastikan pengguna dapat menambahkan barang ke wishlist.	Mengetikkan command WISHLIST ADD dilanjutkan dengan nama barang.	WISHLIST ADD	Barang berhasil ditambahkan ke wishlist, dan sistem memperbarui daftar wishlist pengguna.	Sesuai yang diharapkan.
13	Fitur WISHLIST SWAP <i> <j>	Memastikan pengguna dapat menukar posisi dua barang di wishlist.	Mengetikkan command WISHLIST SWAP dilanjutkan dengan posisi barang pertama dan kedua yang ingin ditukar.	WISHLIST SWAP <i> <j>	Posisi barang pada wishlist berhasil ditukar sesuai input pengguna.	Sesuai yang diharapkan.
14	Fitur WISHLIST REMOVE <i>	Memastikan pengguna dapat menghapus barang dengan posisi tertentu dari wishlist.	Mengetikkan command WISHLIST REMOVE dilanjutkan dengan angka posisi barang tersebut berada	WISHLIST REMOVE <i>	Barang dengan posisi yang dipilih berhasil dihapus, atau sistem akan menampilkan pesan jika barang tidak ditemukan dan masukan posisi angka yang kita input itu tidak valid.	Sesuai yang diharapkan.

15	Fitur WISHLIST REMOVE	Memastikan pengguna dapat menghapus barang tertentu dari wishlist.	Mengetikkan command WISHLIST REMOVE dilanjutkan dengan nama barang yang ingin dihapus.	WISHLIST REMOVE	Barang berhasil dihapus dari wishlist, atau sistem menampilkan pesan jika barang tidak ditemukan.	Sesuai yang diharapkan.
16	Fitur WISHLIST CLEAR	Memastikan pengguna dapat menghapus seluruh barang dari wishlist.	Mengetikkan command WISHLIST CLEAR.	WISHLIST CLEAR	Wishlist berhasil dikosongkan dan menampilkan pesan konfirmasi.	Sesuai yang diharapkan.
17	Fitur WISHLIST SHOW	Memastikan pengguna dapat melihat isi wishlist.	Mengetikkan command WISHLIST SHOW.	WISHLIST SHOW	Daftar barang di wishlist ditampilkan dengan format yang rapi.	Sesuai yang diharapkan.

8 Pembagian Kerja dalam Kelompok

Nama Lengkap / NIM	Deskripsi Tugas
Izhar Alif Akbar / 18223129	Command Load, Command Start, ADT, Melakukan <i>debugging</i> terhadap seluruh fungsi, Memperbaiki dan mengintegrasikan seluruh fungsi ke program Main.
Harfhan Ikhtiar Ahmad R. / 18223123	Command Save, Command Cart Add, Command Wishlist Add, Menyusun Laporan
Stefany Josefina Santono / 18223116	Command Profile, Command Wishlist Clear, Command Wishlist Show, Menyusun Laporan
Nakeisha Valya Shakila / 18223133	Command Wishlist Swap <i> <j>, Command Wishlist Remove <i>, Command Wishlist Remove, Menyusun Laporan
Sharon Darma Putra / 18223107	Command Cart Remove <nama> <n>, Command Cart Show, Daftar ADT, Debbuging Cart Add dan Wishlist

9 Lampiran

9.1 Deskripsi Tugas Besar

PURRMART adalah sebuah aplikasi simulasi berbasis CLI (Command-Line Interface). Sistem ini dibuat menggunakan bahasa C dengan memanfaatkan struktur data yang telah dipelajari selama perkuliahan. Struktur data yang digunakan meliputi ADT Kustom, ADT Stack, ADT Setmap, dan Linked List, yang dimanfaatkan untuk mengelola barang, pengguna, serta berbagai fitur aplikasi. PURRMART juga memungkinkan penggunaan file konfigurasi untuk membaca data awal sistem seperti daftar barang dan pengguna. Library yang diperbolehkan hanya **stdio.h**, **stdlib.h**, **math.h**, dan **time.h**.

System Mechanic

1. About the System

PURRMART adalah sebuah aplikasi yang dapat mensimulasikan aktivitas beli barang pada e-commerce. PURRMART memiliki beberapa fitur utama, yaitu:

- Menampilkan barang toko
- Meminta dan menyuplai barang baru ke toko
- Menyimpan dan membeli barang dalam keranjang
- Menampilkan barang yang sudah dibeli
- Membuat dan menghapus wishlist
- Bekerja untuk menghasilkan uang

2. Menu Program

Ketika program pertama kali dijalankan, PURRMART akan memperlihatkan *main menu* yang berisi *welcome menu* dan beberapa *command* yaitu **START**, **LOAD**, dan juga **HELP**. Setelah itu, program akan memasuki *login menu* yang memiliki command **LOGIN**, **REGISTER**, dan juga **HELP**. Jika pengguna berhasil memasuki kredensial suatu akun, maka mereka akan masuk ke menu selanjutnya. **Main menu** menerima masukan berupa *command* yang akan dijelaskan pada bagian berikutnya. Program akan terus menerima *command* sampai diberikan *command* **QUIT** yang berlaku pada seluruh menu.

3. Command

- **PROFILE** : Melihat data diri pengguna. PROFILE hanya dapat dipanggil saat status pengguna telah login
- **CART ADD <nama> <n>** : Menambahkan barang dengan kuantitas tertentu ke dalam keranjang belanja

- **CART REMOVE <nama> <n>** : Mengurangi barang sejumlah kuantitas tertentu dari keranjang belanja. Perlu dilakukan validasi terhadap kuantitas yang diberikan, bila kuantitas pada keranjang belanja lebih sedikit dari N maka perintah akan gagal
- **CART SHOW** : Menunjukkan barang-barang yang sudah dimasukkan ke dalam keranjang
- **CART PAY** : Membeli barang-barang yang sudah dimasukan ke dalam keranjang. Perlu dipastikan bahwa pengguna memiliki uang yang cukup untuk membeli seluruh barang keranjang. Pembelian akan mengurangi uang yang dimiliki pengguna dan menambahkan riwayat pembelian
- **HISTORY <n>** : Menunjukan riwayat pembelian seorang pengguna. N merupakan jumlah riwayat yang ditampilkan, contoh N=3 maka akan menampilkan 3 riwayat pembelian terbaru. Jika N melebihi jumlah riwayat pembelian yang ada, maka seluruh riwayat pembelian akan ditampilkan. Urutan penunjukan adalah dari yang paling baru ke paling tua.
- **WISHLIST ADD** : Menambahkan suatu barang ke wishlist
- **WISHLIST SWAP <i> <j>** : Menukar barang posisi ke-i dengan barang posisi ke-j pada wishlist. Posisi i dan j merupakan urutan barang pada wishlist, urutan dimulai dari 1
- **WISHLIST REMOVE <i>** : Menghapus barang dengan posisi ke-i dari wishlist
- **WISHLIST REMOVE** : Menghapus barang dari wishlist berdasarkan nama barang yang dimasukkan pengguna
- **WISHLIST CLEAR** : Menghapus semua barang yang terdapat di dalam wishlist
- **WISHLIST SHOW** : Menunjukkan barang-barang yang sudah dimasukkan ke dalam wishlist

4. Perubahan Command

- **START, LOAD, dan SAVE** : Perubahan konfigurasi yang harus ditambahkan dalam implementasi command START, LOAD, dan SAVE.
- **STORE LIST** : Menampilkan nama barang yang dijual beserta harganya.

Konfigurasi Sistem

File konfigurasi akan dibaca saat memulai permainan. File ini menyimpan data-data yang disimpan ketika sistem dijalankan sebelumnya atau data-data default. Spesifikasi dari file konfigurasi adalah sebagai berikut:

1. Barisan pertama adalah bilangan bulat positif **N** yang menunjukkan banyaknya barang di dalam sistem
2. Selanjutnya, sejumlah **N** baris menyatakan nama barang beserta harganya dengan format **<Harga barang> <Nama barang>**
3. Baris selanjutnya adalah bilangan bulat positif **M** yang menunjukkan banyaknya pengguna di dalam sistem
4. Selanjutnya, terdapat data **M** buah pengguna dengan masing-masing spesifikasi berikut:
 - a. Baris pertama adalah bilangan bulat positif **K** yang menunjukkan banyaknya riwayat pengguna tersebut

- b. Selanjutnya, sejumlah **K** baris menyatakan nama barang beserta total biaya dengan format **<Total biaya> <Nama barang>**
5. Baris selanjutnya adalah bilangan bulat positif **J** yang menunjukkan banyaknya wishlist pengguna tersebut
6. Selanjutnya, sejumlah **J** baris menyatakan nama barang dengan format **<Nama barang>**

5. Daftar ADT

- **ADT Kustom** : Perubahan pada ADT pengguna (User). ADT akan menyimpan keranjang, riwayat pembelian, dan wishlist yang dimiliki oleh seorang user.
- **ADT Stack** : Menyimpan riwayat pembelian seorang pengguna. Riwayat pembelian akan menyimpan nama barang dengan ketentuan di bawah dan total biaya seluruh barang.
- **ADT Setmap** : Menyimpan keranjang pembelian seorang pengguna. Keranjang akan menyimpan nama beserta kuantitas barang yang ingin dibeli. Key elemen dalam setmap berupa barang yang pasti unik sementara value adalah kuantitas dari elemen tersebut.
- **ADT Linked List** : Menyimpan wishlist seorang pengguna. Elemen yang disimpan di dalam wishlist adalah nama barang yang ingin dibeli.

6. Bonus

- **Store List Gacor** : Pada iterasi sebelumnya, barang yang masuk ke toko harus bersifat unique. Sebelumnya, kalian menggunakan ADT List Dinamis untuk menyimpan barang-barang tersebut. Apakah terdapat ADT lain yang lebih efisien untuk mengimplementasikan fungsionalitas tersebut? Jika ada, ubah kode store kalian untuk menggunakan ADT-nya.
Catatan: Insertion dan deletion dengan kompleksitas $\sim O(1)$ akan diberikan nilai tambahan.
- **Riwayat Maksimal** : Sistem riwayat pembelian kini menyimpan detail lengkap setiap pembelian, bukan hanya nama barang dengan total harga terbesar. Setiap riwayat mencakup jumlah barang yang dibeli, total harga pembelian, serta rincian harga dan kuantitas per barang. Misalnya, perintah HISTORY 3 akan menampilkan tiga pembelian terakhir dengan rincian lengkap. Format konfigurasi file juga diubah, dengan setiap elemen riwayat diawali jumlah barang dan total harga, diikuti detail barang yang dibeli dalam format **<Total biaya> <Kuantitas> <Nama barang>**.
- **Deteksi Kebocoran Senjata Biologis** : Pada milestone sebelumnya, sistem mendeteksi kode DNA untuk mencegah sabotase musuh. Namun, OWCA mencurigai kebocoran senjata biologis di gudang PURRMART akibat penyimpanan yang tidak sesuai prosedur. Untuk menyelidikinya, OWCA melakukan analisis metagenomik yang membutuhkan sequence alignment antara sekuens senjata biologis dan sekuens metagenomik. Jika skor alignment lebih besar dari 80% panjang sekuens yang lebih panjang, dianggap ada kebocoran senjata biologis. Anda diminta mengimplementasikan algoritma

Needleman-Wunsch untuk global alignment dengan ketentuan scoring: Match (+1), Mismatch (0), Gap penalty (-1).

- **Optimasi Rute Ekspedisi** : Toko PURRMART memiliki banyak klien, dan rekan mereka, SiLambat, membutuhkan cara efisien untuk mengirim barang dengan rute terbaik. Setiap titik harus dikunjungi, dan setiap titik memiliki jarak tertentu ke titik lainnya. Awalnya, algoritma BFS digunakan, namun kurang efisien. Anda diminta untuk menggunakan algoritma yang lebih efisien untuk menyelesaikan masalah ini, dengan penjelasan mengenai alasan pemilihan algoritma dalam laporan.
- **Pencarian Barang** : Toko PURRMART memiliki terlalu banyak barang, dan mesin pencari lama yang digunakan tidak efektif karena hanya mencari barang dengan nama yang persis sama. Untuk mengatasi masalah ini, Anda diminta untuk membuat mesin pencari kuasi match menggunakan jarak Levenshtein, dengan threshold distance default sebesar 10, yang bisa diubah sesuai kebutuhan.

9.2 Notulen Rapat





**Form Asistensi Tugas Besar
IF2111/Algoritma dan Struktur Data STI
Sem. 1 2024/2025**




No. Kelompok/Kelas : 06 / 03
Nama Kelompok : 6acor
Anggota Kelompok (Nama/NIM) :
1. Izhar Alif Akbar / 18223129
2. Harfhan Ikhtiar Ahmad R. / 18223123
3. Stefany Josefina Santono / 18223116
4. Nakeisha Valya Shakila / 18223133
5. Sharon Darma Putra / 18223107
6. Anggita Najmi Layali / 18223122

Asisten Pembimbing : Jonathan Arthurito Aldi Sinaga

Asistensi II

Tanggal :	Catatan Asistensi: Profile -> Isinya sebebass mungkin boleh saja semisalnya menampilkan semua yang ada.Contohnya isi wishlist dan isi cart Masalah segmentation fault yang sebelumnya dihadapi berhasil diselesaikan dengan mengimplementasikan alokasi memori dinamis menggunakan malloc pada struktur data seperti Stack dan SetMap. Solusi ini memungkinkan
Tempat : Google Meetings	

<p>Kehadiran Anggota Kelompok:</p> <p>1 18223129</p>  <p>2 18223123</p>  <p>3 18223116</p>  <p>4 18223133</p>  <p>5 18223107</p>	<p>pengelolaan memori yang lebih fleksibel, sehingga mencegah terjadinya kesalahan pada program saat menangani data yang besar.</p> <p>Asisten meminta untuk memperhatikan case untuk fungsi CART ADD & CART REMOVE agar dapat handle barang yang memiliki angka. Hal ini mencakup penanganan nama barang yang terdiri dari beberapa kata serta validasi angka untuk memastikan input sesuai elemen yang ada.</p> <p>Driver setiap file diperbarui agar mendukung fitur baru dengan efisiensi yang lebih baik. Penyesuaian dilakukan untuk memastikan bahwa setiap driver dapat berjalan sesuai spesifikasi tanpa perlu merubah-ubah konfigurasi yang ada. Contohnya pada driver save & load, terdapat kendala saat mencoba melakukan pengetesan. Yaitu program driver tidak bisa membuka folder DATA.</p>
---	--

 6 18223122 	
	Tanda Tangan Asisten: 

9.3 Log Activity Anggota Kelompok

No.	Log Activities	NIM	Tanggal
1.	Diskusi Konsep dan Pembagian Tugas	18223129 18223123 18223116 18223133 18223107 18223122	14 - 12 - 2024
2.	Push Store List, Cart Pay, dan History	18223122	16 - 12 - 2024
3.	Push ADT Setmap dan ADT Stack	18223107	16 - 12 - 2024

4.	Menambahkan fungsi PrintWishList dan InsertLastWishList pada ADT LinkedList	18223129	17 - 12 - 2024
5.	Push Load dan Start	18223129	17 - 12 - 2024
6.	Push Wishlist Add dan Cart Add	18223123	17 - 12 - 2024
7.	Push Save	18223123	17 - 12 - 2024
8.	Finalisasi ADT	18223107 18223129	17 - 12 - 2024
9.	Push Profile, Wishlist Remove , dan Wishlist Show	18223116	18 - 12 - 2024
10.	Push Wishlist Remove, Wishlist Remove <i>, dan Wishlist Swap	18223133	18 - 12 - 2024
11.	Memperbaiki dan melengkapi Profile, CartAdd, CartRemove, Profile, WishlistAdd, Wishlist Remove, WishlistRemoveIdx, History, Save, dan penyesuaian Login dengan konfigurasi baru	18223129 18223107	21- 12 - 2024
12.	Finalisasi Main	18223129	22 - 12 - 2024
13.	Finalisasi Laporan	18223116 18223123 18223133	22 - 12 - 2024