

# **LAPORAN TUGAS BESAR**

## **IF2111 Algoritma dan Struktur Data STI**

### **PURRMART**

Dipersiapkan oleh:

Kelompok 11

Mochamad Ikhbar A / 18223050

Derick Amadeus Budiono / 18223090


Indana Aulia Ayundazulfa / 18223100

Wilson / 18223012

Naila Selvira Budiana / 18223018

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>IF2111-TB-K02-11</i>		<i>&lt;jml hlm&gt;</i>
		<i>Revisi</i>	<i>&lt;no revisi&gt;</i>	<i>25-11-2024</i>

# Daftar Isi

1	Ringkasan	3
2	Penjelasan Tambahan Spesifikasi Tugas	4
2.1	Quantum Wordl	4
3	Struktur Data (ADT)	4
3.1	Mesin Karakter	4
3.1.1	Sketsa Struktur Data	4
3.1.2	Persoalan yang Diselesaikan	5
3.1.3	Alasan Pemilihan	5
3.1.4	Implementasi	6
3.2	Mesin Kata	6
3.2.1	Sketsa Struktur Data	6
3.2.2	Persoalan yang Diselesaikan	8
3.3	List Dinamis	9
3.3.1	Sketsa Struktur Data	9
3.3.2	Persoalan yang Diselesaikan	11
3.3.3	Alasan Pemilihan	11
3.4	Queue	12
3.4.1	Sketsa Struktur Data	12
3.4.2	Persoalan yang Diselesaikan	14
3.4.3	Alasan Pemilihan	14
3.5.	Array	14
3.5.1.	Sketsa Struktur Data	14
3.5.2.	Persoalan yang Diselesaikan	17
3.5.3.	Alasan Pemilihan	17
4	Program Utama	18
5	Algoritma-Algoritma Menarik	18
5.1	Algoritma Random	18
6	Data Test	19
6.1	Data Test START	19
6.2	Data Test LOAD	19
6.3	Data Test LOAD	20
6.4	Data Test REGISTER	20
6.5	Data Test LOGIN	20
6.6	Data Test STORE LIST	21

6.7 Data Test STORE REQUEST	21
6.8 Data Test STORE SUPPLY	22
6.9 Data Test STORE REMOVE	23
6.10 Data Test WORK	23
6.11 Data Test WORK CHALLENGE	24
7 Test Script	24
8 Pembagian Kerja dalam Kelompok	26
9 Lampiran	27
9.1 Deskripsi Tugas Besar	28
9.2 Notulen Rapat	30
9.3 Log Activity Anggota Kelompok	30

## Ringkasan

Kesulitan yang dialami Agen Purry ketika harus menyuplai segala kebutuhan yang sumbernya sangat sulit dijangkau, borma bojongsoang, membuat agen purry serta OWCA kewalahan. Maka dari itu, tim OWCA menghubungi kelompok 11 untuk dibuatkan sebuah sistem jual beli dengan nama PURRMART.

PURRMART adalah sebuah aplikasi berbasis CLI ( Command-Line Interface ) yang dibuat dengan bahasa C dengan bantuan struktur data terkait list ( statis dan dinamis ), mesin karakter, mesin kata, dan queue. Tak hanya itu, user yang masuk tentunya terautentikasi dan dapat melakukan minigames lainnya selain melihat toko, seperti work dan work challenge. Yang spesial dari tugas besar ini adalah tantangan untuk murni menggunakan mesin kata serta mesin karakter dan tidak diperbolehkannya implementasi scanf dan fgets.

Laporan ini berisikan mengenai penjelasan lebih detail mengenai fitur fitur atau ADT yang kami gunakan saat pengerjaan program PURRMART ini, tes data, dan script yang dilakukan, pembagian kerja, serta lampiran yang terkait.

## 1 Penjelasan Tambahan Spesifikasi Tugas

### 1.1 Riwayat Maksimal

Riwayat Maksimal adalah fitur bonus berupa optimasi dari fitur utama history, yang intinya dapat menampilkan detail pembelanjaan user mulai dari kuantitas barang, total harga per barang, dan total harga keseluruhan tiap payment. Untuk fitur ini, kami mengimplementasikan *separator* yang dapat membatasi informasi tiap payment ( barang kuantitas dan total harga ).

### 1.2 Optimasi Rute

Optimasi rute ekspedisi adalah fitur penerapan dari materi graph *Traveling Salesman Problem*. Dikarenakan spesifikasi mengharuskan menggunakan DFS sebagai metodenya, maka metode pencarian tree dilakukan hingga ujung node terlebih dahulu. Penyelesaian dilakukan dengan metode greedy approach.

## 2 Struktur Data (ADT)

### 2.1 LinkedList

#### 2.1.1 Sketsa Struktur Data

```
#ifndef Linkedlist_H
#define Linkedlist_H

#include "../mesinkata/mesinkata.h"
#include "../mesinkarakter/mesinkarakter.h"
#include "../boolean.h"

#define Nil NULL

typedef Word Infotype;
typedef struct node* Address;
typedef struct node {
    Infotype info;
    Address next;
} Node;
typedef struct {
    Address First;
} Linkedlist;

#define Index(p) (p)->index;
#define Info(p) (p)->info
#define Next(p) (p)->next
#define First(L) ((L).First)

boolean isEmpty(Linkedlist L);

void CreateEmpty(Linkedlist *L);

Address Alokasi(Infotype X);

void Dealokasi(Address *P);

Address Search (Linkedlist L, Infotype X);

void InsVFirst (Linkedlist *L, Infotype X);

void InsVLast (Linkedlist *L, Infotype X);

void DelP (Linkedlist *L, Infotype X);

void PrintInfo (Linkedlist L);
```

```
#endif
```

### 2.1.2 Persoalan yang Diselesaikan

ADT LinkedList sangat terpakai di salah satu function, yaitu Wishlist dimana function tersebut menyimpan list barang yang diinginkan oleh user. Wishlist nantinya akan berhubungan dengan ADT user dan fungsi store.

### 2.1.3 Alasan Pemilihan

Dalam implementasinya, Linked List berperan sangat penting dalam mengatur wishlist dari user, dan juga primitif-primitif yang tersedia sangat berguna untuk digunakan sebagai pengecekan/iterasi lebih lanjut mengenai input yang dimasukkan.

## 2.2 *Mesin Kata*

### 2.2.1 Sketsa Struktur Data

```
/* File: mesinkata.h */
/* Definisi Mesin Kata: Model Akuisisi Versi I */

#ifndef __MESINKATA_H__
#define __MESINKATA_H__

#include "../boolean.h"
#include "../mesinkarakter/mesinkarakter.h"

#define NMax 50
#define BLANK ' '

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;

void IgnoreBlanks();
/* Mengabaikan satu atau beberapa BLANK
   I.S. : currentChar sembarang
   F.S. : currentChar ≠ BLANK atau currentChar = MARK */

void STARTWORD();
/* I.S. : currentChar sembarang
   F.S. : EndWord = true, dan currentChar = MARK;
          atau EndWord = false, currentWord adalah kata yang
sudah diakuisisi,
          currentChar karakter pertama sesudah karakter
terakhir kata */

void ADVWORD();
/* I.S. : currentChar adalah karakter pertama kata yang akan
diakuisisi
   F.S. : currentWord adalah kata terakhir yang sudah
diakuisisi,
          currentChar adalah karakter pertama dari kata
berikutnya, mungkin MARK
          Jika currentChar = MARK, EndWord = true.
   Proses : Akuisisi kata menggunakan procedure SalinWord */

void CopyWord();
/* Mengakuisisi kata, menyimpan dalam currentWord
   I.S. : currentChar adalah karakter pertama dari kata
   F.S. : currentWord berisi kata yang sudah diakuisisi;
          currentChar = BLANK atau currentChar = MARK;
          currentChar adalah karakter sesudah karakter
terakhir yang diakuisisi.
          Jika panjang kata melebihi NMax, maka sisa kata
"dipotong" */

```

```

boolean isEndWord();
/* Mengembalikan true jika EndWord = true */

void printWord(Word Kata);
/* I.S. : Kata terdefinisi
   F.S. : Kata tercetak di layar tanpa karakter tambahan di
   awal maupun di akhir */

boolean StringCompare(Word kata1, Word kata2);
/* Mengembalikan true jika kata1 sama dengan kata2 */

Word str2Word(char* String);
/* Mengubah string menjadi Word */

char* Word2str(Word Kata);
/* Mengubah Word menjadi string */

int Word2int(Word Kata);
/* Mengubah Word menjadi integer */

Word int2Word(int Angka);
/* Mengubah integer menjadi Word */

Word CloneWord(Word kata);
/* Mengembalikan salinan dari kata */

char isOnWord(Word kata, char c);
/* Mengembalikan true jika c terdapat pada kata */

void STARTINPUT ();
/* fungsi dapat memulai sebuah input */

void IgnoreRest();
/* membuat batasan akuisisi kata berdasarkan batasan tertentu
*/

```



```
#endif
```

## 2.2.2 Persoalan yang Diselesaikan

Mesin Kata merupakan pengembangan dari mesin karakter. Tujuan digunakannya mesin kata adalah untuk menerima input user yang berupa sebuah kalimat. Berbeda dengan mesin karakter, pada ADT mesinkata, hal yang menjadi elemen utamanya adalah sebuah kata yang terdiri dari beberapa karakter.

## 2.3 Map

### 2.3.1 Sketsa Struktur Data

```
#ifndef map_H
#define map_H
#include <stdio.h>
#include "../boolean.h"
#include "../barang/barang.h"

/* MODUL Map
Deklarasi stack yang dengan implementasi array eksplisit-statik
rata kiri
*/

// #define false 0
// #define true 1
#define NilMap 0
#define MaxEl 100
#define Undefined -999

// typedef int bool;
typedef CurrentBarang keytype;
typedef int valuetype;
typedef int address;
```

```

typedef struct {
    keytype Key;
    valuetype Value;
} infotype;

typedef struct {
    infotype Elements[MaxEl];
    address Count;
} Map;

/* Definisi Map M kosong : M.Count = NilMap */
/* M.Count = jumlah element Map */
/* M.Elements = tempat penyimpanan element Map */

/* ***** Prototype ***** */

/* *** Konstruktor/Kreator *** */
void CreateEmptyMap(Map *M);
/* I.S. Sembarang */
/* F.S. Membuat sebuah Map M kosong berkapasitas MaxEl */
/* Ciri Map kosong : count bernilMapai NilMap */

/* ***** Predikat Untuk test keadaan KOLEKSI ***** */
boolean IsEmptyMap(Map M);
/* Mengirim true jika Map M kosong*/
/* Ciri Map kosong : count bernilMapai NilMap */

boolean IsFullMap(Map M);
/* Mengirim true jika Map M penuh */
/* Ciri Map penuh : count bernilMapai MaxEl */

/* ***** Operator Dasar Map ***** */
valuetype ValueMap(Map M, keytype k);
/* Mengembalikan nilMapai value dengan key k dari M */
/* Jika tidak ada key k pada M, akan mengembalikan Undefined */

void InsertMap(Map *M, keytype k, valuetype v);

```

```

/* Menambahkan Elmt sebagai elemen Map M. */
/* I.S. M mungkin kosong, M tidak penuh
    M mungkin sudah beranggotakan v dengan key k */
/* F.S. v menjadi anggota dari M dengan key k. Jika k sudah ada,
    operasi tidak dilakukan */

void DeleteMap(Map *M, keytype k);
/* Menghapus Elmt dari Map M. */
/* I.S. M tidak kosong
    element dengan key k mungkin anggota / bukan anggota dari
    M */
/* F.S. element dengan key k bukan anggota dari M */

boolean IsMemberMap(Map M, keytype k);
/* Mengembalikan true jika k adalah member dari M */

void PrintMap(Map M);

void ubahValueMap(Map *M, keytype k, valuetype v);

#endif

```

### 2.3.2 Persoalan yang Diselesaikan

Map digunakan pada fungsi cart. Fungsi cart memiliki fitur untuk menambahkan, mengurangi/menghilangkan barang dan juga membayar. Pada cart itu sendiri, implementasinya sama seperti dunia nyata yang mencari barang berdasarkan nama nya, bukan melalui indeks keberapanya.

### 2.3.3 Alasan Pemilihan

Alasan pemilihan map sebagai fungsi pada cart adalah karena pada nyatanya, pemilihan barang akan dilihat berdasarkan namanya. Seperti pada nyatanya, isi dari cart pasti selalu berantakan (Tidak tersusun).

## 2.4 *Stack*

### 2.4.1 Sketsa Struktur Data

```

/* File : stack.h */
/* deklarasi stack yang diimplementasi dengan tabel kontigu dan
ukuran sama */
/* TOP adalah alamat elemen puncak */
/* Implementasi dalam bahasa C dengan alokasi statik */
#ifndef stackt_H
#define stackt_H

#include "../boolean.h"
#include "../barang/barang.h"

#define NilStack -1
#define MaxEl 100
/* NilStack adalah stack dengan elemen kosong . */

typedef int infotypestack;
typedef int address; /* indeks tabel */

typedef struct {
    CurrentBarang item;
    int quantity;
    int total_harga;
} CartItem;

/* Contoh deklarasi variabel bertipe stack dengan ciri TOP : */
/* Versi I : dengan menyimpan tabel dan alamat top secara eksplisit*/
typedef struct {
    CartItem items[20];
    infotypestack T[MaxEl]; /* tabel penyimpan elemen */
    address TOP; /* alamat TOP: elemen puncak */
} Stack;

/* Definisi stack S kosong : S.TOP = NilStack */
/* Elemen yang dipakai menyimpan nilStackai Stack T[0]..T[MaxEl-1] */
/* Jika S adalah Stack maka akses elemen : */
/* S.T[(S.TOP)] untuk mengakses elemen TOP */
/* S.TOP adalah alamat elemen TOP */

/* Definisi akses dengan Selektor : Set dan Get */
#define Top(S) (S).TOP

```

```

#define InfoTop(S) (S).T[(S).TOP]

/* ***** Prototype ***** */
/* *** Konstruktor/Kreator *** */
void CreateEmptyStack(Stack *S);
/* I.S. sembarang; */
/* F.S. Membuat sebuah stack S yang kosong berkapasitas MaxEl */
/* jadi indeksanya antara 0.. MaxEl */
/* Ciri stack kosong : TOP bernilStackai NilStack */

void initStack(Stack *s);
/* ***** Predikat Untuk test keadaan KOLEKSI ***** */
boolean IsEmptyStack(Stack S);
/* Mengirim true jika Stack kosong: lihat definisi di atas */
boolean IsFullStack(Stack S);
/* Mengirim true jika tabel penampung nilStackai elemen stack penuh */

/* ***** Menambahkan sebuah elemen ke Stack ***** */
void Push(Stack * S, CartItem X);
/* Menambahkan X sebagai elemen Stack S. */
/* I.S. S mungkin kosong, tabel penampung elemen stack TIDAK penuh */
/* F.S. X menjadi TOP yang baru, TOP bertambah 1 */

/* ***** Menghapus sebuah elemen Stack ***** */
void Pop(Stack * S, CartItem* X);
/* Menghapus X dari Stack S. */
/* I.S. S tidak mungkin kosong */
/* F.S. X adalah nilStackai elemen TOP yang lama, TOP berkurang 1 */

void PrintStack(Stack S);
    /* I.S. Stack S terdefinisi */
    /* F.S. Elemen-elemen dalam stack dicetak dari TOP ke bawah dengan
format [elemen1, elemen2, ..., elemenN] */

int LengthStack (Stack S);

#endif

```

### 2.4.2 Persoalan yang Diselesaikan

ADT Stack sangat terpakai di salah satu function store, yaitu riwayat\_pembelian dimana function tersebut memungkinkan untuk mencatat histori pembelian yang dilakukan oleh user.

### 2.4.3 Alasan Pemilihan

Alasan Stack digunakan dibanding ADT lainnya karena sejatinya fitur history adalah riwayat pembelian barang yang juga mengharuskan stack digunakan dalam implementasinya.

## 3 Program Utama

Program utama dimulai dengan include modul modul yang diperbolehkan serta include semua ADT yang terpakai di program ini. Terdapat 2 modul utama yang terpakai yaitu <stdio.h> dan <stdlib.h> dan semua ADT file yang berjumlah 11 ADT, yaitu barang.h, delay.h, linked\_list.h, list\_dinamis.h, map.h, mesinkarakter.h, mesinkata.h, queue.h, random.h, stack.h, dan store.h.

Masuk ke bagian utama program, pertama tama dimulai dengan deklarasi queue, list, linked\_list, dan list dinamis sebagai container pertama terhadap segala elemen, baik mengenai user ataupun barang dalam store. Lalu dilanjut dengan looping *while(true)* agar sebuah looping terus berjalan hingga user ingin menyudahinya, di dalam blok loop terdapat pengkondisian dengan command yang mungkin diinput oleh user beserta pemanggilan masing masing fungsi yang terkait dengan command.

Terdapat beberapa command utama, yaitu START untuk memulai sesi, LOAD untuk menginput file konfigurasi, REGISTER untuk membuat akun baru dengan username yang unik, LOGIN untuk masuk ke program sebagai sebuah akun untuk menggunakan fitur fitur spesifik lainnya, STORE LIST untuk menampilkan barang yang tersedia di toko, STORE REQUEST untuk input barang baru ke antrean supply barang, STORE SUPPLY untuk memasukkan barang baru ke list store, WORK memungkinkan user mendapatkan uang sebagai imbalan dari melakukan sebuah pekerjaan, WORK CHALLENGE memungkinkan user memainkan mini games yang imbalannya adalah uang, terdapat 3 mini games utama yaitu WORDL3, TEBAK ANGKA, dan QUANTUM WORDL3, PROFILE untuk melihat data pengguna, CART ADD untuk menambahkan barang ke dalam keranjang belanja, CART REMOVE untuk mengurangi barang tertentu dari keranjang belanja, CART SHOW untuk menunjukkan isi keranjang belanja, CART PAY untuk membeli barang-barang yang sudah ada dalam keranjang, HISTORY untuk menunjukkan riwayat pembelian, WISHLIST ADD untuk menambah barang ke wishlist, WISHLIST SWAP untuk mengubah urutan wishlist, WISHLIST remove untuk menghapus barang dari wishlist, WISHLIST CLEAR untuk menghapus semua barang dari wishlist, WISHLIST SHOW untuk menunjukkan isi dari wishlist, LOGOUT untuk keluar dari program sebagai sebuah akun, SAVE untuk menyimpan segala konfigurasi yang terjadi, baik update barang atau update user, dan QUIT untuk benar benar keluar dari program.

## 4 Algoritma-Algoritma Menarik

### 4.1 Algoritma Random

Algoritma Random merupakan salah satu algoritma yang menarik untuk diterapkan karena algoritma random ini menggunakan fungsi time yang merujuk pada waktu saat ini sebagai seed dari random yang akan digunakan. Algoritma random ini digunakan dalam beberapa spesifikasi seperti spesifikasi tebak angka dan spesifikasi wordl3.

**Algoritma :**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "random.h"
#include "../mesinkata/mesinkata.h"

int Random() {
    int max_num = 100;
    srand(time(0));

    int random = (rand() % max_num) + 1;

    return random;
}

char* katarandom(const char *words[], int wordCount) {
    // Map the random number (1 to 100) to a valid index (0 to wordCount - 1)
    int randomIndex = (Random() - 1) % wordCount;

    // Explicitly cast away the const qualifier
    return (char*)words[randomIndex];
}
```

## 4.2 Algoritma Graph

Algoritma ini merupakan salah satu algoritma yang menarik dan diterapkan pada fitur OPTIMASIRUTE. Algoritma yang digunakan termasuk greedy graph. implementasi Optimasi rute pun termasuk kedalam implementasi TSP (Travelling Salesman Problem ) menggunakan pendekatan DFS untuk membangun pohon pencarian semua rute yang mungkin dirutekan dan dapat dipastikan memakan cost minimum. TSP adalah masalah optimasi yang mencari rute minimum untuk mengunjungi semua node dalam graf tepat sekali, kemudian kembali ke node awal. Masalah ini menggunakan:

- **Graf tak berarah** dengan representasi **adjacency matrix**.
- **DFS Tree Traversal** untuk menjelajahi semua kemungkinan rute.

```
#include <stdio.h>
# include "graph.h"
# include "../ADT/mesinkarakter/mesinkarakter.h"
# include "../ADT/mesinkata/mesinkata.h"

int graph[MAX][MAX];    // Graph adjacency matrix
int visited[MAX];        // Array for visited nodes
int bestPath[MAX];       // Best path storage
int tempPath[MAX];       // Temporary path storage
int minCost = INF;       // Minimum cost initialized to INF
int currentCost = 0;     // Current traversal cost
int n;                   // Number of nodes

/* Fungsi untuk menampilkan path */
void printPath(int path[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", path[i]);
    }
    printf("\n");
}

/* Fungsi untuk membuat pohon DFS dan mengecek semua rute */
void buildTree(int current, int depth) {
    visited[current] = 1;    // Tandai node sebagai dikunjungi
```



```

    tempPath[depth - 1] = current; // Simpan node saat ini dalam path
    sementara

    // Jika sudah mengunjungi semua node, kembali ke node awal
    if (depth == n) {
        if (graph[current][0] != INF) { // Cek jika ada jalur kembali ke
0
            currentCost += graph[current][0]; // Tambahkan biaya kembali
ke awal
            tempPath[depth] = 0; // Tambahkan node awal ke akhir
path

            // Cek apakah ini path dengan biaya minimum
            if (currentCost < minCost) {
                minCost = currentCost;
                for (int i = 0; i <= n; i++) {
                    bestPath[i] = tempPath[i]; // Simpan path terbaik
                }
            }

            // Cetak setiap path yang selesai (optional for tree
visualization)
            printf("Path ditemukan: ");
            printPath(tempPath, n + 1);
            currentCost -= graph[current][0]; // Backtrack biaya
        }
        visited[current] = 0; // Batalkan tanda kunjungan
        return;
    }

    // Cek semua node yang bisa menjadi anak dari node saat ini
    (pembangunan pohon)
    for (int i = 0; i < n; i++) {
        if (!visited[i] && graph[current][i] != INF) {
            currentCost += graph[current][i]; // Tambahkan biaya
            buildTree(i, depth + 1); // Rekursi ke node anak
            currentCost -= graph[current][i]; // Backtrack biaya

```

```

    }
}

visited[current] = 0; // Batalkan tanda kunjungan (backtrack)
}

/* Fungsi untuk mengecek input duplikat */
int isDuplicateEdge(int u, int v) {
    return graph[u][v] != INF; // Jika sudah ada jalur, kembalikan true
}

void OptimasiRute() {
    int edges, i, u, v, w;

    printf("Masukkan jumlah lokasi pengiriman (node): ");
    STARTWORD();
    if (isKataInteger(currentWord)) {
        n = Word2int(currentWord);
    } else {
        printf("Input tidak valid! Silakan masukkan angka.\n");
        return;
    }

    printf("Masukkan jumlah rute (edge): ");
    STARTWORD();
    if (isKataInteger(currentWord)) {
        edges = Word2int(currentWord);
    } else {
        printf("Input tidak valid! Silakan masukkan angka.\n");
        return;
    }

    /* Inisialisasi matriks graph dengan nilai INF */
    for (i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {

```

```

        if (i == j)
            graph[i][j] = 0; // Jarak ke diri sendiri adalah 0
        else
            graph[i][j] = INF;
    }
}

/* Masukkan data rute */
printf("Masukkan jarak antarlokasi (format: asal tujuan jarak):\n");
for (i = 0; i < edges; i++) {
    while (1) { // Loop sampai input valid
        STARTWORD();
        u = Word2int(currentWord);
        ADVWORD();
        v = Word2int(currentWord);
        ADVWORD();
        w = Word2int(currentWord);

        // Cek duplikat atau edge terbalik
        if (u < 0 || u >= n || v < 0 || v >= n) {
            printf("Node tidak valid! Silakan ulangi.\n");
        } else if (isDuplicateEdge(u, v) || isDuplicateEdge(v, u)) {
            printf("Jalur antara %d dan %d sudah ada! Silakan
masukkan rute yang berbeda:\n", u, v);
        } else {
            graph[u][v] = w;
            graph[v][u] = w; // Karena graf tidak berarah
            break;
        }
    }
}

/* Inisialisasi array visited */
for (i = 0; i < n; i++) {
    visited[i] = 0;
}

```

```

printf("Membangun pohon pencarian, silakan tunggu...\n");

/* Bangun pohon dari node 0 */
buildTree(0, 1);

/* Cetak hasil akhir */
printf("\nRute paling efektif adalah: ");
for (i = 0; i <= n; i++) {
    printf("%d ", bestPath[i]);
}
printf("\nBiaya minimum: %d\n", minCost);
}

```

## 5 Data Test

### 5.1 Data Test PROFILE

Fitur yang dites : PROFILE  
 Hasil yang diharapkan :

```

>> PROFILE
Username: user1
Password: password1
Anda telah login ke PURRMART sebagai user1.
Username: user1
Uang: 1000
Riwayat Pembelian:
Cart:
Wishlist:

```

Hasil :

```
Username: DER
Uang: 100
Riwayat:
Riwayat kosong.
Cart:
Cart is empty.
Wishlist:
[]
```

```
Username: NAILA
Uang: 8670
Riwayat:
Isi riwayat:
1. AYAM 1 1200
2. MIE AYAM 13 130
Cart:
MIE AYAM: 2
Wishlist:
[AYAM]
```

## 5.2 Data Test CART ADD

Fitur yang ditest : CART ADD

Hasil yang diharapkan :

```
>> CART ADD AYAM GORENG 10
//test case store memiliki ayam goreng
AYAM GORENG berhasil ditambahkan sebanyak 10

>>CART ADD AYAM KUKUS 10
//test case ayam kukus tidak ada di store
barang tidak ada di store

>>CART ADD AYAM GORENG 5
//test case menambahkan barang yang sudah ada di cart
AYAM GORENG berhasil ditambahkan sebanyak 5
```

Hasil :

//test case store memiliki ayam goreng

```
>> CART ADD AYAM GORENG 10
AYAM GORENG berhasil ditambahkan sebanyak 10
```

//test case store tidak memiliki ayam kukus

```
>> STORE LIST
List barang yang ada di toko:
- AK 47 - Harga: 10
- AYAM GORENG - Harga: 10
```

```
>> CART ADD AYAM KUKUS 10
Barang tidak ada di store
```

//test case menambahkan barang ke cart

```
>> CART ADD AYAM GORENG 5  
AYAM GORENG berhasil ditambahkan kembali sebanyak 5
```

Isi Keranjang:

Jumlah	Nama Barang	Harga
15	AYAM GORENG	150

### 5.3 Data Test CART REMOVE

Fitur yang ditest : CART REMOVE

Hasil yang diharapkan :

```
>> CART REMOVE AYAM GORENG 2  
// test case menghilangkan barang di cart dengan n < total barang di  
cart
```

Hasil :

Sebelum Penghapusan :

Isi Keranjang:

Jumlah	Nama Barang	Harga
3	AYAM GORENG	30
2	AK47	94

```
AYAM GORENG berhasil dihapus sebanyak 2  
=====[PURRMART LOBBY]=====
```

Setelah penghapusan :

Isi Keranjang:

Jumlah	Nama Barang	Harga
1	AYAM GORENG	10
2	AK47	94

## 5.4 Data Test CART SHOW

Fitur yang ditest : CART SHOW

Hasil yang diharapkan :

```
>>CART SHOW
Isi Keranjang:
Jumlah      | Nama Barang          | Harga
-----
5           | AYAM GORENG          | 50
2           | AK47                  | 94
```

Hasil :

```
Isi Keranjang:
Jumlah      | Nama Barang          | Harga
-----
5           | AYAM GORENG          | 50
2           | AK47                  | 94
-----[BURPMART LOBBY]-----
```

## 5.5 Data Test CART PAY

Fitur yang ditest : CART PAY

Hasil yang diharapkan :

```
>> Cart Pay
Kamu akan membeli barang-barang berikut :
Isi Keranjang:
Jumlah      | Nama Barang          | Harga
-----
50          | Apple goreng         | 250000
50          | Banana kukus         | 100000
20          | Orange mantap        | 60000

Total biaya yang harus dikeluarkan adalah 410000. Apakah jadi dibeli ? (
Ya/Tidak ) : Ya
Selamat kamu telah membeli barang-barang tersebut !

>> Cart Pay
Kamu akan membeli barang-barang berikut :
```

Isi Keranjang:

Jumlah	Nama Barang	Harga
--------	-------------	-------

50	Apple goreng	250000
50	Banana kukus	100000
20	Orange mantap	60000

Total biaya yang harus dikeluarkan adalah 410000. Apakah jadi dibeli ? ( Ya/Tidak ) : Tidak  
Pembelian dibatalkan

>> Cart Pay

Kamu akan membeli barang-barang berikut :

Isi Keranjang:

Jumlah	Nama Barang	Harga
--------	-------------	-------

50	Apple goreng	250000
50	Banana kukus	100000
20	Orange mantap	60000

Total biaya yang harus dikeluarkan adalah 410000. Apakah jadi dibeli ? ( Ya/Tidak ) : Purry  
Pembelian dibatalkan

>> Cart Pay

Kamu akan membeli barang-barang berikut :

Isi Keranjang:

Jumlah	Nama Barang	Harga
--------	-------------	-------

50	Apple goreng	250000
50	Banana kukus	100000
20	Orange mantap	60000

Total biaya yang harus dikeluarkan adalah 410000. Apakah jadi dibeli ? ( Ya/Tidak ) : Ya  
Transaksi gagal! saldo kamu hanya 100, sedangkan total harga adalah 410000.

//test case cart pay “Ya” dan berhasil

>> Cart Pay

Kamu akan membeli barang-barang berikut:

Isi Keranjang:

Jumlah	Nama Barang	Harga
--------	-------------	-------

50	Apple goreng	250000
50	Banana kukus	100000
20	Orange mantap	60000

Total biaya yang harus dikeluarkan adalah 410000. Apakah jadi dibeli? (Ya/Tidak): Ya  
Selamat kamu telah membeli barang-barang tersebut!



//test case cart pay “Tidak” dan pembelian batal

```
>> Cart Pay
Kamu akan membeli barang-barang berikut:
Isi Keranjang:
Jumlah      | Nama Barang      | Harga
-----
50          | Apple goreng     | 250000
50          | Banana kukus     | 100000
20          | Orange mantap    | 60000

Total biaya yang harus dikeluarkan adalah 410000. Apakah jadi dibeli? (Ya/Tidak): Tidak
Pembelian dibatalkan.
```

//test case cart pay “Ya” tetapi uang tidak cukup

```
>> Cart Pay
Kamu akan membeli barang-barang berikut:
Isi Keranjang:
Jumlah      | Nama Barang      | Harga
-----
50          | Apple goreng     | 250000
50          | Banana kukus     | 100000
20          | Orange mantap    | 60000

Total biaya yang harus dikeluarkan adalah 410000. Apakah jadi dibeli? (Ya/Tidak): Ya
Transaksi gagal! Saldo kamu hanya 100, sedangkan total harga adalah 410000.
```

// Test case input purry

```
>> Cart Pay
Kamu akan membeli barang-barang berikut:
Isi Keranjang:
Jumlah      | Nama Barang      | Harga
-----
50          | Apple goreng     | 250000
50          | Banana kukus     | 100000
20          | Orange mantap    | 60000

Total biaya yang harus dikeluarkan adalah 410000. Apakah jadi dibeli? (Ya/Tidak): Purry
Pembelian dibatalkan.
```

## 5.6 Data Test HISTORY

Fitur yang ditest : HISTORY

Hasil yang diharapkan :

```
>> HISTORY 1
// test case riwayat pembelian kosong
Kamu belum membeli barang apapun !

>> HISTORY 1
//test case riwayat pembelian terisi satu history
Pembelian 1 - Total 410000
Kuantitas  | Nama Barang      | Total Harga
-----
20          | Orange mantap    | 60000
50          | Banana kukus     | 100000
```

50	Apple goreng	250000
----	--------------	--------

>> HISTORY 2

//test case riwayat pembelian terdiri dari 2 pesanan

Pembelian 1 - Total 410000

Kuantitas	Nama Barang	Total Harga
-----------	-------------	-------------

20	Orange mantap	60000
50	Banana kukus	100000
50	Apple goreng	250000

Pembelian 2 - Total 2800

Kuantitas	Nama Barang	Total Harga
-----------	-------------	-------------

2	Orange	800
5	Banana	1500
5	Apple	500

>> HISTORY 1

//test case riwayat pembelian terdiri dari 2 pesanan

Pembelian 1 - Total 410000

Kuantitas	Nama Barang	Total Harga
-----------	-------------	-------------

20	Orange mantap	60000
50	Banana kukus	100000
50	Apple goreng	250000

>> HISTORY 10

//test case riwayat pembelian terdiri dari 2 pesanan

Pembelian 1 - Total 410000

Kuantitas	Nama Barang	Total Harga
-----------	-------------	-------------

20	Orange mantap	60000
50	Banana kukus	100000
50	Apple goreng	250000

Pembelian 2 - Total 2800

Kuantitas	Nama Barang	Total Harga
-----------	-------------	-------------

2	Orange	800
5	Banana	1500
5	Apple	500

//test case user belum membeli barang apapun

```
>> History 1
Kamu belum membeli barang apapun!
```

//test case terdapat satu riwayat pembelian

```
>> History 1
Riwayat pembelian barang:

Pembelian 1 - Total 410000
Kuantitas | Nama Barang | Total Harga
-----|-----|-----
20         | Orange mantap | 60000
50         | Banana kukus  | 100000
50         | Apple goreng  | 250000
```

// test case terdapat tepat 2 riwayat pembelian

```
>> History 2
Riwayat pembelian barang:

Pembelian 1 - Total 410000
Kuantitas | Nama Barang | Total Harga
-----|-----|-----
20         | Orange mantap | 60000
50         | Banana kukus  | 100000
50         | Apple goreng  | 250000

Pembelian 2 - Total 2800
Kuantitas | Nama Barang | Total Harga
-----|-----|-----
2         | Orange      | 800
5         | Banana      | 1500
5         | Apple       | 500
```

//test case n < riwayat

```
Isi riwayat:
1. END_TRANSACTION 3 2800
2. Orange 2 800
3. Banana 5 1500
4. Apple 5 500
5. END_TRANSACTION 3 410000
6. Orange mantap 20 60000
7. Banana kukus 50 100000
8. Apple goreng 50 250000
>> History 1
Riwayat pembelian barang:

Pembelian 1 - Total 410000
Kuantitas | Nama Barang | Total Harga
-----|-----|-----
20         | Orange mantap | 60000
50         | Banana kukus  | 100000
50         | Apple goreng  | 250000
```

// test case n > riwayat

```
Isi riwayat:
1. END_TRANSACTION 3 2800
2. Orange 2 800
3. Banana 5 1500
4. Apple 5 500
5. END_TRANSACTION 3 410000
6. Orange mantap 20 60000
7. Banana kukus 50 100000
8. Apple goreng 50 250000
>> History 10
Riwayat pembelian barang:

Pembelian 1 - Total 410000
Kuantitas | Nama Barang | Total Harga
-----|-----|-----
20         | Orange mantap | 60000
50         | Banana kukus  | 100000
50         | Apple goreng  | 250000

Pembelian 2 - Total 2800
Kuantitas | Nama Barang | Total Harga
-----|-----|-----
2         | Orange      | 800
5         | Banana      | 1500
5         | Apple       | 500
```

## 5.7 Data Test WISHLIST ADD

Fitur yang dites : WISHLIST ADD

Hasil yang diharapkan :

```
>> WISHLIST ADD
Masukkan nama barang: Tes 1

Berhasil menambahkan Tes 1 Crispy Besthal ke wishlist!
//test case barang ada di store

>> WISHLIST ADD
Masukkan nama barang: Tes 1

Tes 1 sudah ada di wishlist
//test case barang sudah ada di wishlist

>> WISHLIST ADD
Masukkan nama barang: Tes 5

Tidak ada barang dengan nama Tes 5!
//test case barang tidak ada di store
```

```
>> WISHLIST ADD
Masukkan nama barang : Tes 1
Berhasil menambahkan Tes 1 ke wishlist!
```

//test case barang sudah ada di wishlist

```
>> WISHLIST ADD
Masukkan nama barang : Tes 1
Tes 1 sudah ada di wishlist!
```

//test case barang tidak ada di store

```
>> WISHLIST ADD
Masukkan nama barang : Tes 5
Tidak ada barang dengan nama Tes 5!
```

## 5.8 Data Test WISHLIST SWAP

Fitur yang ditest : WISHLIST SWAP

Hasil yang diharapkan :

```
>> WISHLIST SWAP 1 2
Berhasil menukar posisi Tes 1 dengan Tes 2 pada wishlist!
//test case barang 1 dan 2 berhasil ditukar

>> WISHLIST SWAP 1 5
Penukaran gagal, Barang ke-5 tidak ada di wishlist!
//test case hanya terdapat satu barang

>> WISHLIST SWAP 1 2
Wishlist kosong!
//test case wishlist kosong
```

Hasil

//test case barang 1 dan 2 berhasil ditukar

```
>> WISHLIST SWAP 1 2
Berhasil menukar posisi Tes 1 dengan Tes 2 pada wishlist!
```

//test case hanya terdapat satu barang

```
>> WISHLIST SWAP 1 5
Penukaran gagal, Barang ke-5 tidak ada di wishlist
```

//test case wishlist kosong :

```
>> WISHLIST SWAP 1 2
Wishlist kosong!
```

## 5.9 Data Test WISHLIST REMOVE

Fitur yang ditest : WISHLIST REMOVE

Hasil yang diharapkan :

```
>> WISHLIST REMOVE 1
Berhasil menghapus barang posisi ke-2 dari wishlist!
//test case berhasil menghapus barang
```

```
>> WISHLIST REMOVE 10
Penghapusan barang WISHLIST gagal dilakukan, Barang ke-10 tidak ada di
WISHLIST!
//test case jumlah barang tidak mencukupi

>> WISHLIST REMOVE 1
Penghapusan barang WISHLIST gagal dilakukan, WISHLIST kosong!
//test case wishlist kosong

>> WISHLIST REMOVE
Masukkan nama barang yang akan dihapus : Tes 1
Tes 1 berhasil dihapus dari WISHLIST!
//test case remove menggunakan nama

>> WISHLIST REMOVE
Masukkan nama barang yang akan dihapus : Tes 2
Penghapusan barang WISHLIST gagal dilakukan, Tes 2 tidak ada di WISHLIST!
//test case barang tidak ada di wishlist
```

Hasil

//test case berhasil menghapus barang

```
>> WISHLIST REMOVE 1
Berhasil menghapus barang posisi ke-1 dari wishlist!
```

//test case jumlah barang tidak mencukupi

```
>> WISHLIST REMOVE 10
Penghapusan barang wishlist gagal dilakukan, Barang ke-10 tidak ada di wishlist!
```

//test case wishlist kosong

```
>> WISHLIST REMOVE 1
Penghapusan barang wishlist gagal dilakukan, wishlist kosong!
```

//test case remove menggunakan nama

```
>> WISHLIST REMOVE
Masukkan nama barang yang ingin dihapus : Tes 1
Tes 1 berhasil dihapus dari wishlist
```

//test case barang tidak ada di wishlist

```
>> WISHLIST REMOVE
```

Masukkan nama barang yang ingin dihapus : Tes 2

Penghapusan barang WISHLIST gagal dilakukan, Tes 2 tidak ada di wishlist

### 5.10 Data Test WISHLIST CLEAR

Fitur yang ditest : WISHLIST CLEAR

Hasil yang diharapkan :

```
>> WISHLIST CLEAR
```

Wishlist telah dikosongkan.

```
//test case mengosongkan wishlist
```

Hasil

```
>> WISHLIST CLEAR
```

wishlist telah dikosongkan.

### 5.11 Data Test WISHLIST SHOW

Fitur yang ditest : WISHLIST SHOW

Hasil yang diharapkan :

```
>> WISHLIST SHOW
```

Berikut adalah isi wishlist-mu:

1 Tes 1

2 Tes 2

3 Tes 3

4 Tes 4

```
//test case wishlist tidak kosong
```

```
>> WISHLIST SHOW
```

Wishlist kamu kosong!

```
//test case wishlist kosong
```

Hasil

```
//test case wishlist tidak kosong
```

```
>> WISHLIST SHOW
Berikut ini adalah wishlist-mu:
1 Tes 1
2 Tes 2
3 Tes 3
4 Tes 4
```

//test case wishlist kosong

```
>> WISHLIST SHOW
Wishlist kamu kosong!
```

## 6 Test Script

Isi dengan skenario test yang dimungkinkan untuk semua fitur yang ada. Bisa dibuat dalam bentuk tabel sebagai berikut:

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	PROFILE	Memeriksa apakah fungsi dapat menampilkan data pengguna	Pengguna memasukkan command "PROFILE".	TEST PROFILE	Fungsi menampilkan data pengguna.	Fungsi menampilkan data pengguna.
2	CART ADD	Memeriksa apakah fungsi dapat menambah barang ke keranjang belanja.	Pengguna memasukkan command "CART ADD <nama> <n>".	TEST CART ADD	Fungsi dapat menambah barang ke keranjang belanja.	Fungsi dapat menambah barang ke keranjang belanja.
3	CART REMOVE	Memeriksa apakah fungsi dapat mengurangi barang sejumlah kuantitas tertentu dari keranjang belanja.	Pengguna memasukkan command "CART REMOVE <nama> <n>".	<a href="#">TEST CART REMOVE</a>	Fungsi dapat mengurangi barang sejumlah kuantitas tertentu dari keranjang belanja.	Fungsi dapat mengurangi barang sejumlah kuantitas tertentu dari keranjang belanja.
4	CART SHOW	Memeriksa apakah fungsi dapat menunjukkan barang-barang yang terdapat dalam keranjang.	Memasukkan command "CART SHOW"	<a href="#">TEST CART SHOW</a>	Fungsi dapat menunjukkan barang-barang yang terdapat dalam keranjang.	Fungsi dapat menunjukkan barang-barang yang terdapat dalam keranjang.
5	CART PAY	Memeriksa apakah fungsi dapat mengurangi uang yang	Memasukkan command "CART PAY"	<a href="#">TEST CART PAY</a>	Fungsi dapat mengurangi uang yang dimiliki	Fungsi dapat mengurangi uang yang dimiliki



		dimiliki pengguna dan menambahkan riwayat pembelian.			pengguna dan menambahkan riwayat pembelian.	pengguna dan menambahkan riwayat pembelian.
6	HISTORY	Memeriksa apakah fungsi dapat menunjukkan n buah riwayat pembelian.	Memasukkan command "HISTORY <n>"	<a href="#">TEST HISTORY</a>	Fungsi dapat menunjukkan n buah riwayat pembelian.	Fungsi dapat menunjukkan n buah riwayat pembelian.
7	STORE SUPPLY	Memeriksa apakah function dapat benar benar memasukkan barang baru di antrean dengan beberapa command spesifik seperti TERIMA, TUNDA, TOLAK, dan PURRY	Memasukkan command "STORE SUPPLY"	<a href="#">TEST STORE SUPPLY</a>	Fungsi dapat menampilkan HEAD OF QUEUE dari barang yang akan disupply ke store list. Pengecekan dimulai secara bertahap dari command ke command ( Terima, Tolak, Purry, dan Tunda ).	Fungsi dapat menampilkan HEAD OF QUEUE dari barang yang akan disupply ke store list. Pengecekan dimulai secara bertahap dari command ( Terima, Tolak, Purry, dan Tunda ).
8	WISHLIST ADD	Memeriksa apakah fungsi dapat menambahkan barang ke wishlist.	Pengguna memasukkan perintah "WISHLIST ADD".	<a href="#">TEST WISHLIST ADD</a>	Fungsi dapat menambahkan barang ke wishlist.	Fungsi dapat menambahkan barang ke wishlist.
9	WISHLIST SWAP	Memeriksa apakah fungsi dapat menukar urutan 2 barang dalam wishlist.	Pengguna akan memasukkan command "WISHLIST SWAP".	<a href="#">TEST WISHLIST SWAP</a>	Fungsi dapat menukar urutan 2 barang dalam wishlist.	Fungsi dapat menukar urutan 2 barang dalam wishlist.
10	WISHLIST REMOVE	Memeriksa apakah fungsi dapat menghapus barang dari wishlist.	Pengguna akan memasukkan command "WISHLIST REMOVE <i>" atau "WISHLIST REMOVE"	<a href="#">TEST WISHLIST REMOVE</a>	Fungsi dapat menghapus barang dari wishlist.	Fungsi dapat menghapus barang dari wishlist.
11	WISHLIST CLEAR	Memeriksa apakah fungsi dapat mengosongkan wishlist.	Pengguna akan memasukkan command "WISHLIST CLEAR".	<a href="#">TEST WISHLIST CLEAR</a>	Fungsi dapat mengosongkan n wishlist.	Fungsi dapat mengosongkan wishlist.
12	WISHLIST SHOW	Memeriksa apakah fungsi dapat menampilkan isi wishlist.	Pengguna akan memasukkan command "WISHLIST SHOW".	<a href="#">TEST WISHLIST SHOW</a>	Fungsi dapat Menampilkan isi wishlist.	Fungsi dapat menampilkan isi wishlist.

## 7 Pembagian Kerja dalam Kelompok

Nama Lengkap - NIM	Deskripsi Tugas
Mochamad Ikhbar Adiwinangun - 18223050	<ul style="list-style-type: none"> <li>- Mengerjakan fitur store dan membuat struct barang</li> <li>- Mengerjakan ADT queue, list dinamis, dan membuat tambahan fungsi minor di mesin kata</li> <li>- Mengerjakan laporan</li> </ul>
Derick Amadeus Budiono - 18223090	<ul style="list-style-type: none"> <li>- Mengerjakan cart add, cart remove dan show cart</li> <li>- membuat fitur bonus optimasi rute</li> <li>- Membuat laporan sesuai dari yang dikerjakan</li> </ul>
Indana Aulia Ayundazulfa - 18223100	<ul style="list-style-type: none"> <li>- Mengerjakan fitur load dan save</li> <li>- Mengerjakan laporan</li> </ul>
Wilson - 18223012	<ul style="list-style-type: none"> <li>- Mengerjakan fitur wishlist remove, wishlist remove &lt;i&gt;, wishlist show, dan wishlist swap, serta mengintegrasikannya ke main</li> <li>- Mengerjakan laporan</li> </ul>
Naila Selvira Budiana - 18223018	<ul style="list-style-type: none"> <li>- Mengerjakan fitur wishlist add, wishlist clear, dan profile</li> <li>- Merevisi adt user</li> <li>- Mengerjakan laporan</li> </ul>
Rahmat Pujiyanto	<ul style="list-style-type: none"> <li>- Ilang</li> </ul>

## 8 Lampiran

### 8.1 Deskripsi Tugas Besar

#### - Spesifikasi Umum

Buatlah sebuah aplikasi simulasi berbasis CLI (*command-line interface*). Sistem ini dibuat dalam **bahasa C** dengan menggunakan **struktur data yang sudah kalian pelajari** di mata kuliah ini. Kalian boleh menggunakan (atau memodifikasi) struktur data yang sudah kalian buat untuk praktikum pada tugas besar ini. Daftar ADT yang wajib digunakan dapat dilihat pada bagian Daftar ADT. *Library* yang boleh digunakan hanya **stdio.h**, **stdlib.h**, **time.h**, dan **math.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

Pengguna dapat memasukkan *command-command* berikut.

**a. START**

START merupakan salah satu *command* yang dimasukkan pertama kali dalam Toko PURRMART. Setelah menekan Enter, dibaca file konfigurasi *default* yang berisi daftar barang pada toko.

**b. LOAD <filename>**

LOAD merupakan salah satu *command* yang dimasukkan pertama kali dalam PURRMART. Command ini memiliki satu argumen yaitu *filename* yang merepresentasikan suatu *save file* yang ingin dibuka. *File* didapatkan dari *folder* tertentu, contohnya *save*. Setelah menekan *Enter*, akan dibaca *save file <filename>* yang berisi daftar barang pada toko. Lebih detailnya bisa dilihat pada Konfigurasi Aplikasi.

**c. LOGIN**

Login merupakan *command* yang baru dapat dipanggil setelah pengguna memulai sesi. *Login* berguna untuk masuk ke akun di sistem PURRMART yang sudah didaftarkan sebelumnya.

**d. LOGOUT**

LOGOUT merupakan salah satu *command* yang baru dapat digunakan setelah pengguna telah memasuki sebuah sesi.

**e. REGISTER**

Register merupakan *command* yang baru dapat dipanggil setelah pengguna memulai sesi. *Register* berguna untuk mendaftarkan akun baru ke dalam sistem PURRMART. Sebuah akun setidaknya memiliki atribut *username* dan *password*. **Username dan password hanya terdiri dari 1 kata.**

**f. WORK**

WORK merupakan *command* yang digunakan pengguna untuk mendapatkan uang. Terdapat sejumlah pekerjaan yang bisa dipilih. Setiap pekerjaan memiliki waktu tunggu yang berbeda-beda dan dengan nominal pendapatan yang berbeda-beda pula. Selama pengguna sedang bekerja, maka sistem tidak bisa digunakan hingga pekerjaan selesai dilakukan.

**g. WORK CHALLENGE**

WORK CHALLENGE merupakan *command* alternatif sebagai cara mendapatkan uang dengan melakukan *challenge-challenge* di OWCA. Pemain membutuhkan uang dengan jumlah tertentu untuk bisa memainkan challenge. Uang yang dibayarkan untuk bermain *challenge* tidak akan dikembalikan, meskipun pemain kalah dalam permainan. Terdapat dua *challenge* yang dapat dipilih:

**a) Tebak Angka**

Challenge Tebak Angka merupakan permainan yang meminta pemain menebak sebuah angka yang ditentukan oleh program. Pemain memiliki 10 (sepuluh) kesempatan untuk menebak angka yang benar. Program akan memberikan *feedback* apakah angka tebakannya lebih besar, lebih kecil, atau sama dengan angka target. Jumlah kesempatan yang dipakai oleh pengguna akan mempengaruhi uang yang didapatkan.

**b) WORDL3**

Challenge WORDL3 merupakan permainan tebak kata berjumlah lima karakter. Pemain memiliki 6 (enam) kesempatan untuk menebak kata yang benar. Kata harus berupa kata valid, tidak boleh sekadar *string* acak, bahasa dibebaskan (disarankan bahasa Indonesia/Inggris). Pada setiap giliran, program akan mencetak ulang kata yang dimasukkan, tetapi dengan penanda tertentu. Huruf yang benar dan berada pada tempat yang tepat diberi tanda “!char!”. Huruf yang benar, tetapi berada di tempat yang salah diberi tanda “(char)” setelah hurufnya. Huruf yang tidak ada sama sekali pada kata diberi tanda “|char|” setelah hurufnya.

**h. STORE LIST**

STORE LIST adalah *command* yang digunakan untuk melihat barang-barang apa saja yang ada di dalam toko. **Setiap barang yang ditampilkan haruslah bersifat *unique*.**

**i. STORE REQUEST**

STORE REQUEST adalah *command* yang digunakan untuk meminta penambahan barang baru ke dalam toko. Barang-barang yang diminta akan disimpan di dalam sebuah antrian dan akan dimasukkan ke toko menggunakan *command* selanjutnya. **Nama barang yang masuk tidak boleh sama dengan nama barang yang sudah ada di toko atau di antrian.**

**j. STORE SUPPLY**

STORE SUPPLY adalah *command* yang digunakan untuk menambahkan barang baru ke dalam toko berdasarkan antrian permintaan. Barang yang berada pada antrian paling

depan akan dimasukkan ke toko. Pengguna dapat menerima, menunda, atau menolak permintaan.

- Jika diterima, maka program akan meminta harga dari barang dan dimasukkan ke toko.
- Jika ditunda, maka barang akan kembali masuk ke antrian
- Jika ditolak, maka barang akan dihapus dari antrian

Harus terdapat validasi agar harga barang merupakan angka yang valid (berupa angka dan bernilai lebih dari nol).

#### **k. STORE REMOVE**

STORE REMOVE adalah *command* yang dapat menghapus barang yang ada di toko. Akan dilakukan *input* akan barang yang akan dihapus. Beri tahu apabila proses berhasil (barang terdapat pada toko dan berhasil dihapus) ataupun tidak (barang tidak terdapat di toko).

#### **l. PROFILE**

PROFILE merupakan *command* yang digunakan menampilkan data-data dari pengguna. Data-data yang ditampilkan adalah nama dan saldo pengguna, namun dapat dikreasikan.

#### **m. CART ADD**

CART ADD merupakan *command* yang digunakan untuk menambahkan barang ke keranjang belanja. Akan dilakukan *input* berupa nama barang dan jumlah barang yang ingin dimasukkan, dan akan dikeluarkan output sesuai dengan kondisi yang berlaku.

#### **n. CART REMOVE**

CART REMOVE merupakan *command* yang digunakan untuk mengurangi kuantitas suatu barang tertentu dari keranjang belanja. Akan dilakukan *input* berupa nama barang dan jumlah barang yang ingin dikurangi, dan akan dikeluarkan output sesuai dengan kondisi yang berlaku.

#### **o. CART SHOW**

CART SHOW merupakan *command* yang digunakan untuk menunjukkan barang-barang yang sudah dimasukkan ke dalam keranjang.

#### **p. CART PAY**

CART PAY adalah *command* yang digunakan untuk membeli barang-barang yang sudah dimasukkan 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.

**q. HISTORY**

HISTORY merupakan command yang digunakan untuk menunjukkan riwayat pembelian seorang pengguna. Pengguna dapat memasukkan input berupa angka untuk menunjukkan sejumlah riwayat pembelian terbaru.

**r. WISHLIST ADD**

WISHLIST ADD merupakan command yang digunakan untuk menambahkan suatu barang pada wishlist. Akan dilakukan input berupa nama barang dan akan dikeluarkan output sesuai dengan kondisi yang berlaku.

**s. WISHLIST SWAP**

WISHLIST SWAP merupakan command yang digunakan untuk menukar posisi 2 barang pada wishlist. Akan dilakukan input berupa 2 angka yang akan dijadikan sebagai index barang yang ingin ditukar.

**t. WISHLIST REMOVE**

WISHLIST REMOVE merupakan command yang digunakan untuk menghilangkan suatu barang dari wishlist. Pengguna dapat memilih untuk melakukan input berupa angka sebagai indeks barang yang dihilangkan ataupun memasukkan nama barang yang ingin dihilangkan.

**u. WISHLIST CLEAR**

WISHLIST CLEAR merupakan command yang digunakan untuk mengosongkan wishlist.

**v. WISHLIST SHOW**

WISHLIST SHOW merupakan command yang digunakan untuk menampilkan barang-barang yang sudah dimasukkan dalam wishlist.

**w. HELP**

HELP merupakan *command* yang digunakan menampilkan daftar *command* yang mungkin untuk dieksekusi dengan deskripsinya. Penjelasan dari deskripsi dibebaskan selama masih mendeskripsikan *command* sesuai dengan spek.

**x. SAVE <filename>**

SAVE merupakan *command* yang digunakan untuk menyimpan *state* aplikasi terbaru ke dalam suatu *file*. Command SAVE memiliki satu argumen yang merepresentasikan nama *file* yang akan disimpan. Penyimpanan dilakukan pada *folder* tertentu, misal *folder save*.

y. **QUIT**

**QUIT** merupakan *command* yang digunakan untuk keluar dari sesi aplikasi PURRMART.




## 8.2 Notulen Rapat

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




No. Kelompok/Kelas : 11/K2  
 Nama Kelompok :  
 Anggota Kelompok (Nama/NIM) :  
     1. Derick Amadeus B / 18223090  
     2. Wilson / 18223012  
     3. Mochamad Ikhbar A / 18223050  
     4. Indana Aulia Ayundazulfa / 18223100  
     5. Naila Selvira Budiana / 18223018

Asisten Pembimbing : Jonathan Arthurito Aldi Sinaga


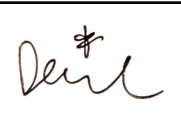




### Asistensi I

<b>Tanggal :</b> 18 November 2024			<b>Catatan Asistensi:</b> Banyak branch yang kalau tidak dipakai dihapus aja. Pastikan ADT yang terpakai di masing-masing fitur sudah di commit di branch khusus supaya fungsi yang digunakan sama dan tidak menyulitkan saat merge.
<b>Tempat :</b> Daring via Gmeets			
<b>Kehadiran Anggota Kelompok:</b>			
No	NIM	Tanda Tangan	
1	18223050		
2	18223090		
3	18223100		







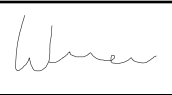

4	18223012		
5	18223018		
			<b>Tanda Tangan Asisten:</b> 

## Asistensi II

<b>Tanggal : 23 November 2024</b>			<b>Catatan Asistensi:</b>  Beri komentar yang sama, rapikan github, dan pastikan tidak ada conflict saat melakukan merge.
<b>Tempat : Daring via Gmeets</b>			
<b>Kehadiran Anggota Kelompok:</b>			
No	NIM	Tanda Tangan	
1	18223050		
2	18223090		
3	18223100		
4	18223012		
5	18223018		
			<b>Tanda Tangan Asisten:</b> 

--	--

### Asistensi III

Tanggal : 17 Desember 2024			<b>Catatan Asistensi:</b>  Banyak branch yang tidak terpakai hapus saja. Fitur fitur sudah aman, tinggal segera di merge
Tempat : Daring via Gmeets			
Kehadiran Anggota Kelompok:			
No	NIM	Tanda Tangan	
1	18223050		
2	18223090		
3	18223100		<b>Tanda Tangan Asisten:</b>  
4	18223012		
5	18223018		

### 8.3 Log Activity Anggota Kelompok

<u>No</u>	<u>Tanggal</u>	<u>NIM</u>	<u>Nama</u>	<u>Aktivitas</u>
1	14/12/2024	18223050 18223090	M Ikhbar A Derick Amadeus Budiono	Membuat ADT stack, MAP, dan fitur cart di branch ibay-derick
2	15/12/2024	18223090	Derick Amadeus Budiono	Fixing bugs cart di branch ibay-derick
3	17/12/2024	18223050	M Ikhbar A	membuat bonus riwayat maksimal di branch ibay-derick
4	18/12/2024	18223050 18223090	M Ikhbar A Derick Amadeus Budiono	Fixing bugs history dan cart di branch ibay-derick
5	19/12/2024	18223050	M Ikhbar A	Update driver stack di branch ibay-derick
6	20/12/2024	18223050	M Ikhbar A	Mengubah bonus history agar sesuai dengan konfigurasi save load di branch ibay-derick
7	16/12/2024	18223012	Wilson	Membuat fitur wishlist di branch Wilson
8	17/12/2024	18223012	Wilson	Update fitur wishlist di branch Wilson
9	17/12/2024	18223018	Naila Selvira	Membuat fitur user profile dan wishlist add di branch naila
10	17/12/2024	18223100	Indana Aulia	Membuat fitur load dan save dengan konfigurasi baru

11	18/12/2024	18223100	Indana Aulia	Update save dan load
12	19/12/2024	18223100	Indana Aulia	Update save dan load
13	20/12/2024	18223100	Indana Aulia	Finalisasi save dan load
14	18/12/2024	18223012	Wilson	Menambahkan fitur wishlist ke branch main
15	19/12/2024	18223 18223090	Wilson Derick Amadeus Budiono	Menambahkan ADT ke branch main dan menambahkan driver linkedlist
16	20/12/2024	18223090 18223018 18223050	Derick Amadeus Budiono Naila Selvira M Ikhbar A	Fixing bugs, update mesinkata
17	21/12/2024	18223100 18223050	Indana Aulia M Ikhbar A	Menambah hiasan di main.c dan fixing load serta save di main.c
18	22/12/2024	18223090	Derick Amadeus Budiono	Final fix untuk membuat makefile serta menambahkan fitur Optimasi Rute di main.c
19	22/12/2024	18223100	Indana Aulia	Membenarkan output dan register