

LAPORAN TUGAS BESAR

IF2111 Algoritma dan Struktur Data STI


PURRMART

Dipersiapkan oleh:

Kelompok 12

Jennifer Grachel Alicia / 18221018
Darren Mansyl / 18223001
Samuel Chris M. B. S. / 18223011
M. Adam Mirza / 18223015
Ferro Arka Berlian / 18223027
Jason Samuel / 18223091

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

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		<i>IF2111-TB-02-12</i>		32
		<i>Revisi</i>	0	25 November 2024

Daftar Isi

1	Ringkasan	4
2	Penjelasan Tambahan Spesifikasi Tugas	5
3	Struktur Data (ADT)	6
3.1	ADT Custom	6
3.2	ADT Stack	8
3.3	ADT Setmap	9
3.4	ADT Linked List	10
4	Program Utama	10
5	Data Test	12
5.1	Data Test Start	12
5.2	Data Test Load	12
5.3	Data Test Save	12
5.4	Data Test Store List	13
5.5	Data Test Profile	13
5.6	Data Test Cart Add	14
5.7	Data Test Cart Remove <nama><n>	15
5.8	Data Test Cart Show	15
5.9	Data Test Cart Pay	16
5.10	Data Test History	17
5.11	Data Test Wishlist Add	18
5.12	Data Test Wishlist Swap	19
5.13	Data Test Wishlist Remove <i>	19
5.14	Data Test Wishlist Remove	20
5.15	Data Test Wishlist Clear	21
5.16	Data Test Wishlist Show	21
6	Test Script	22
7	Pembagian Kerja dalam Kelompok	29
8	Lampiran	30
8.1	Deskripsi Tugas Besar	30
■ 8.1.1	Latar Belakang	30
■ 8.1.2	Spesifikasi Umum	31
■ 8.1.3	System Mechanic	32
1.	About the System	32
2.	Menu Program	32
3.	Command	32
■ 8.1.4	Konfigurasi Sistem	34
■ 8.1.5	Daftar ADT	35

■ 8.1.6 Bonus	36
■ 8.1.7 Catatan Tambahan	38
8.2 Notulen Rapat	39
8.3 Log Activity Anggota Kelompok	41

1 Ringkasan

Agen Purry membutuhkan sistem jual beli ke Borma yang bernama PURRMART karena tidak memiliki transportasi untuk menuju Borma. Sistem tersebut digunakan untuk mendapatkan pasokan barang-barang perang dari Borma. Barang - barang perang tersebut disiapkan untuk nantinya digunakan melawan Dr. Asep Spakbor yang sedang membuat mesin ‘Oppenheimer-inator’ yang akan menghancurkan wilayah tiga negara bagian.

PURRMART merupakan aplikasi jual beli pada *e-commerce* berbasis CLI (Command-line interface). Program dimulai dengan menampilkan *welcome menu* yang berisi *command start* dan *load*. Setelah *start/load*, pengguna akan masuk ke *login menu* dan dapat melakukan *command login, register, dan quit*.

Setelah *login/register*, pengguna memasuki *main menu* yang memiliki *command* utama, yaitu *work, work challenge*, serta beberapa macam *store*, seperti *list, request, supply, dan remove*. Ditambah juga dengan spesifikasi *work challenge*, seperti Tebak Angka dan World3.

Purrmart juga memiliki fitur *cart, history, dan wishlist*. Pada fitur *cart*, pengguna dapat menambah, menghapus, menampilkan, dan membayar daftar barang yang terdapat dalam *cart*. Pada fitur *history*, pengguna dapat menampilkan daftar pembayaran yang telah dilakukan sebelumnya. Pada fitur *wishlist*, pengguna dapat menambahkan, menghapus, menukar urutan, dan menampilkan daftar barang yang berada di dalam *wishlist*.

Secara garis besar, laporan ini menjelaskan mengenai deskripsi umum persoalan sistem, tambahan spesifikasi tugas, struktur data yang digunakan, penjelasan mengenai program utama, algoritma menarik yang didapatkan, data dan script yang digunakan untuk menguji sistem, serta lampiran lainnya, seperti pembagian tugas dan notulensi rapat.

2 Penjelasan Tambahan Spesifikasi Tugas

Tidak ada penjelasan tambahan.

3 Struktur Data (ADT)

Pada pembuatan sistem untuk PURRMART, digunakan beberapa beberapa sketsa data-struktur data (ADT) secara bersamaan sebagai dasar untuk mengatasi persoalan yang ditemukan. Dalam algoritmanya, terdapat 4 jenis ADT, yaitu ADT *Custom*, ADT *List*, ADT Mesin Karakter dan Mesin Kata, serta ADT *Queue*.

3.1 ADT *Custom*

- Sketsa struktur data

```
typedef struct {  
    char nama[100];  
    int pendapatan;  
    int durasi;  
} Pekerjaan
```

- Persoalan yang diselesaikan : Dalam menyelesaikan fitur *work*, suatu pekerjaan merupakan gabungan dari nama pekerjaan, pendapatan pekerjaan, dan durasi pekerjaan.
- Alasan pemilihan : Dibanding dengan memisah nama pekerjaan, pendapatan pekerjaan, dan durasi pekerjaan yang saling kebergantungan ke dalam 3 *array* berbeda yang tidak efisien.
- Implementasi : dideklarasikan sebagai struktur data pekerjaan di *console.h* dan diimplementasikan di pekerjaan sebagai sebuah *list* dari ADT Pekerjaan yang digunakan untuk menyimpan nama pekerjaan, pendapatan pekerjaan, dan durasi pekerjaan.

- Sketsa struktur data :

```
typedef struct {  
    char nama[100];  
    int  
    password[100];  
}
```

```

    int uang;
    Map keranjang;
    Stack
    riwayat_pembelian;
    List wishlist;
} User

```

- Persoalan yang diselesaikan : Dalam fitur *load*, *save*, *register*, *login*, dan *logout*, setiap *user* memiliki *username*/nama, *password*, dan saldo uang. Struktur data ini akan menyimpan informasi user menggunakan *list* statis.
- Alasan pemilihan : Penggunaan lebih sederhana karena memori sudah dialokasikan di awal sehingga tidak perlu proses alokasi dan dealokasi memori. Selain itu, jumlah pengguna juga tidak akan terlalu banyak. Apabila ada penambahan *user*, hanya mengandalkan indeks *array* saja. Dalam fitur *register*, proses pencarian *username* dapat dilakukan secara traversal untuk mengecek apakah *username* sudah digunakan. Jika sudah digunakan, maka akan menemukan kecocokan dengan elemen di dalam list.
- Implementasi : dideklarasikan sebagai struktur data *user* di *kustom.h* dan diimplementasikan di *ListUser* sebagai data yang dibutuhkan untuk membuat *list*-nya.
- Sketsa struktur data

```

typedef struct {
    char name[100];
    int harga;
} Barang;

```

- Persoalan yang diselesaikan : Dalam fitur *load*, *save*, semua *store*, dan *Bioweapon*, setiap barang membutuhkan nama barang dan harga barang. Struktur data ini menyimpan informasi tersebut menggunakan *list* dinamis.

- Alasan pemilihan : Penggunaan struktur data ini menggunakan *list* dinamis karena banyak barang yang bisa ditambahkan dalam satu waktu dan kemungkinan slot dari *array* tersebut bisa penuh, jadi penggunaan *list* dinamis ini untuk mencegah penuhnya slot dan akan bertambah sembari barang ditambahkan (jika penuh)
- Implementasi : Dideklarasikan sebagai struktur data barang di *kustom.h* dan diimplementasikan di *ListBarang* sebagai data yang dibutuhkan untuk membuat *list*-nya.

3.2 ADT Stack

- Sketsa struktur data :

```
typedef struct {
    char* namaBarang;
    int totalHarga;
} infotypeStack;

typedef int
address_stack;

typedef struct {
    infotypeStack
T[MaxElStack];
    address_stack TOP;
} Stack;
```

- Persoalan yang diselesaikan : Dalam fitur *profile* dan *history*, riwayat pembelian membutuhkan nama barang dan total harga dari barang tersebut yang disimpan di struktur data *infotypeStack*. Jika kita sudah selesai membuat/melakukan suatu pembelian, riwayat tersebut akan langsung diletakkan di riwayat teratas sebagai riwayat paling baru.

- Alasan pemilihan : Penggunaan struktur data Stack pada bagian *profile* dan *history* karena struktur data ini menggunakan prinsip *Last In*, dan *First Out*. Struktur data ini memberikan efisiensi waktu dengan hanya mengakses data teratas dari struktur tersebut.
- Implementasi : Dideklarasikan sebagai struktur data *Stack* di *stack.h* dan diimplementasikan di *stack.c*

3.3 ADT Setmap

- Sketsa struktur data:

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

typedef struct {
    infotype
    Elements[MaxElMap];
    address_map Count;
} Map;
```

- Persoalan yang diselesaikan : Dalam semua fitur *cart* (*cartadd*, *cartremove*, *cartshow*, dan *cartpay*), struktur data ini dibutuhkan untuk menyimpan nama sebagai *key* dan harga sebagai *value*.
- Alasan pemilihan : Dengan struktur data ini, akan mencegah adanya redudansi data karena nama yang ada akan selalu unik. Dengan struktur data Map, suatu *cart* akan mudah ditambahkan, dihilangkan, dan di-*update* dengan menggunakan ADT yang sudah tersedia.
- Implementasi : Dideklarasikan sebagai struktur data Map di *map.h* dan diimplementasikan fungsinya sebagai ADT di *map.c*

3.4 ADT Linked List

- Sketsa struktur data :

```
typedef struct
tElmtlist {
    infotypelist
info;
    address_list
next;
} ElmtList;

typedef struct {
    address_list
First;
} List;
```

- Persoalan yang diselesaikan : Dalam semua fitur *wishlist* (*wishlistadd*, *wishlistswap*, *wishlistremoveid*, *wishlistremovenam*, *wishlistclear*, *wishlistshow*), struktur data ini dibutuhkan untuk menempatkan nama barang yang diinginkan di *info* dan menyambungkan ke *wishlist* selanjutnya dengan *next*.
- Alasan pemilihan : Penggunaan struktur data ini karena mudah dalam pengaksesan seperti membuat *wishlist*, pemindahan, pencarian, dan penghapusan.
- Implementasi : Dideklarasikan di *listlinear.h* dan diimplementasikan di *listlinear.c*

4 Program Utama

Program utama dari PURRMART dimulai dengan *main menu* yang menampilkan *welcome menu* dengan *command* START, LOAD, dan HELP. Pengguna dapat memberikan *command* START untuk memulai program tanpa membaca *file* konfigurasi yang telah tersedia, LOAD untuk memulai program sesuai dengan pilihan *file* konfigurasi yang sebelumnya sudah tersedia, dan HELP untuk bantuan. Setelah memasuki *login menu*, terdapat *command* LOGIN, REGISTER,

dan HELP. Setelah pengguna berhasil memasuki kredensial akun, maka mereka akan masuk ke menu selanjutnya, yaitu *main menu*. Untuk keluar dari program, terdapat *command* QUIT untuk digunakan.

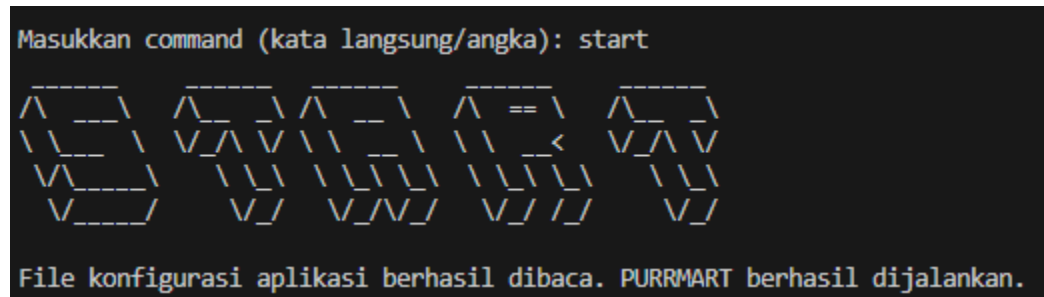
Setelah *file* dibaca, pengguna memasuki *login menu* untuk masuk ke akun. Setelah masuk ke akun (*login*), pengguna dapat melihat data diri dengan memilih *command* PROFILE. Terdapat fitur *Cart* dan *Wishlist* yang dapat diatur oleh pengguna. Jika ingin menambahkan barang ke dalam cart, gunakan *command* CART ADD. Sedangkan, jika pengguna ingin mengurangi barang dengan kuantitas tertentu, gunakan CART REMOVE <nama><n> dimana keranjang belanja harus lebih sedikit dari N. Untuk melihat apa saja yang ada dalam keranjang, maka gunakan *command* CART SHOW. Terakhir, jika pengguna ingin membeli barang-barang yang ada di cart, gunakan *command* CART PAY dengan ketentuan pengguna memiliki uang yang cukup untuk membeli seluruh barang. Untuk melihat riwayat pembelian, pengguna dapat menggunakan *command* HISTORY<n> dimana N merupakan jumlah riwayat pembelian yang ada.

Ada juga fitur *Wishlist* yang dapat ditambah menggunakan *command* WISHLIST ADD. Jika pengguna ingin menukar barang posisi ke-i dengan barang posisi ke-j, maka gunakan *command* WISHLIST SWAP <i><j>. Terdapat 2 macam *remove* untuk menghapus *wishlist*, yaitu WISHLIST REMOVE untuk menghapus barang dari *wishlist* berdasarkan nama barang yang dimasukkan dan WISHLIST REMOVE <i> yang digunakan untuk menghapus barang dengan posisi ke-i dari *wishlist*. Untuk menghapus semua barang yang ada di *wishlist*, gunakan *command* WISHLIST CLEAR. Terakhir, untuk menampilkan barang-barang yang sudah dimasukkan ke dalam *wishlist*, gunakan *command* WISHLIST SHOW. Terdapat juga *command* STORE LIST untuk menampilkan nama barang yang dijual beserta harganya.

5 Data Test

5.1 Data Test Start

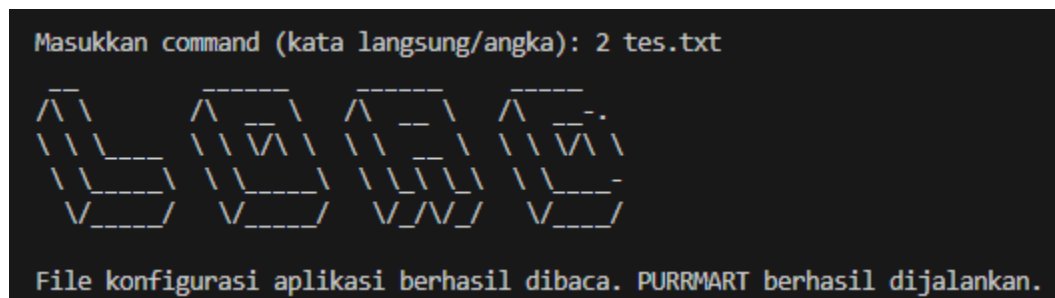
Test dilakukan untuk memastikan keberjalan program utama dan kondisi awal program. Program akan menampilkan ART dan *file* konfigurasi berhasil dibaca.



Gambar 5.1 Berhasil memulai START

5.2 Data Test Load

Test ini memastikan program dapat membaca *save file* yang memiliki riwayat terakhir aplikasi PURRMART.



Gambar 5.2 Berhasil membaca *save file*

5.3 Data Test Save

Test ini memastikan sesi yang kita jalankan tersimpan dalam *file* konfigurasi.

```
Masukkan command (kata langsung/angka): save tes.txt

-----
\_/ \_/ \_/ \_/ \_/
\_/ \_/ \_/ \_/ \_/
\_/ \_/ \_/ \_/ \_/
\_/ \_/ \_/ \_/ \_/

Data berhasil disimpan ke file tes.txt
```

Gambar 5.3 Berhasil menyimpan *file*

5.4 Data Test Store List

Test ini dilakukan untuk menampilkan daftar barang yang ada di toko.

```
Masukkan command (kata langsung/angka): store list

-----
\_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/
\_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/
\_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/

Daftar barang di toko:
1. AK47 Price: 10
2. Lalabu Price: 20
3. Ayam Goreng Crisbar Price: 20
4. Meong Price: 500
5. Kai Cenat Price: 1
6. Satualivin Price: 200
Tekan karakter apapun untuk melanjutkan: |
```

Gambar 5.4 Menampilkan daftar barang di toko

5.5 Data Test Profile

Test ini memastikan bahwa program dapat memperlihatkan data diri pengguna.

```
Masukkan command (kata langsung/angka): profile

=====
\^  == \^  == \^  == \^  == \^  == \^  == \^  ==
\\  -/ \\  -/ \\  -/ \\  -/ \\  -/ \\  -/ \\  -/
\\  -/ \\  -/ \\  -/ \\  -/ \\  -/ \\  -/ \\  -/
V_/_/ V_/_/ V_/_/ V_/_/ V_/_/ V_/_/ V_/_/ V_/_/

Nama: adam
Uang: 61160
Password: ****

Apakah anda ingin melihat password? (y/n):
```

Gambar 5.5 Menampilkan data diri pengguna

5.6 Data Test Cart Add

Test ini memastikan bahwa program dapat menambahkan barang dengan kuantitas tertentu ke keranjang belanja.

```
Masukkan command (kata langsung/angka): cart add AK47 23

=====
\^  == \^  == \^  == \^  == \^  == \^  == \^  ==
\\  -/ \\  -/ \\  -/ \\  -/ \\  -/ \\  -/ \\  -/
\\  -/ \\  -/ \\  -/ \\  -/ \\  -/ \\  -/ \\  -/
V_/_/ V_/_/ V_/_/ V_/_/ V_/_/ V_/_/ V_/_/ V_/_/

Berhasil menambahkan 23 AK47 ke keranjang belanja!
```

Gambar 5.6.1 CART ADD berhasil

```
Masukkan command (kata langsung/angka): cart add alstrup 23

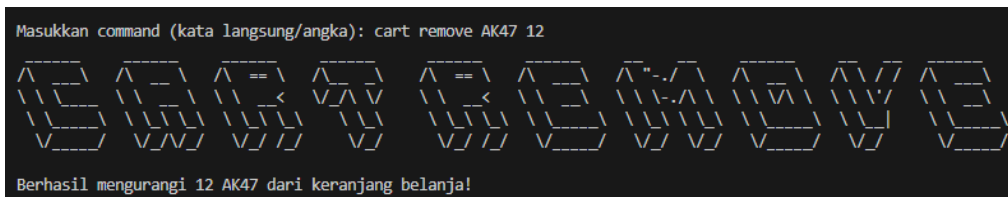
=====
\^  == \^  == \^  == \^  == \^  == \^  == \^  ==
\\  -/ \\  -/ \\  -/ \\  -/ \\  -/ \\  -/ \\  -/
\\  -/ \\  -/ \\  -/ \\  -/ \\  -/ \\  -/ \\  -/
V_/_/ V_/_/ V_/_/ V_/_/ V_/_/ V_/_/ V_/_/ V_/_/

Barang tidak ada di toko!
```

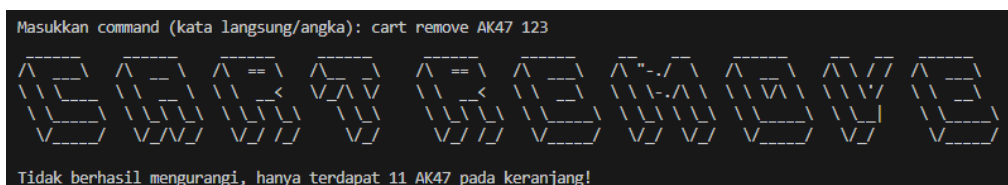
Gambar 5.6.2 CART ADD gagal

5.7 Data Test Cart Remove <nama><n>

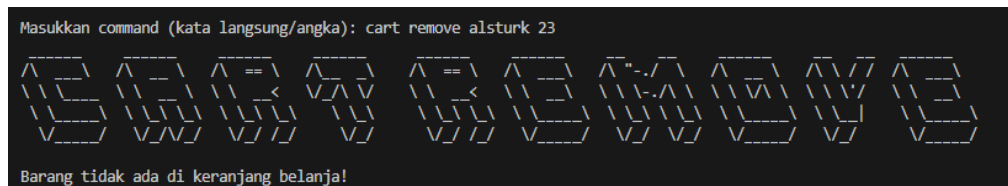
Test ini memastikan program dapat mengurangi barang dengan jumlah kuantitas tertentu dari keranjang belanja dengan ketentuan harus lebih sedikit dari N.



Gambar 5.7.1 Barang ada di keranjang dengan jumlah yang mencukupi



Gambar 5.7.2 Barang ada di keranjang dengan jumlah yang tidak mencukupi



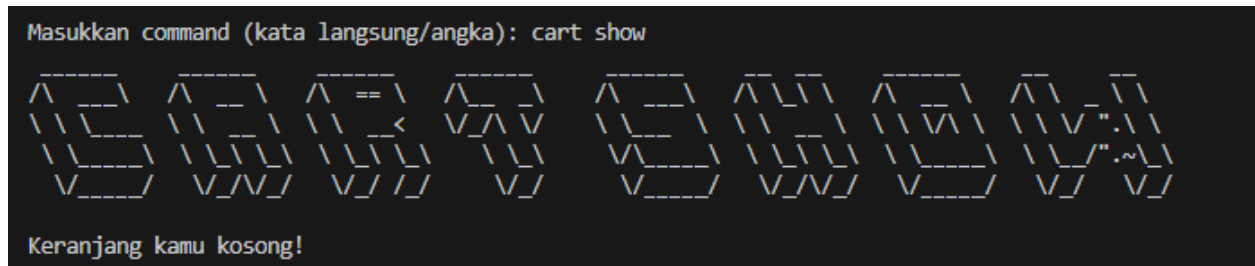
Gambar 5.7.3 Barang tidak ada di keranjang

5.8 Data Test Cart Show

Test ini memastikan program dapat menunjukkan barang yang dimasukkan ke keranjang.



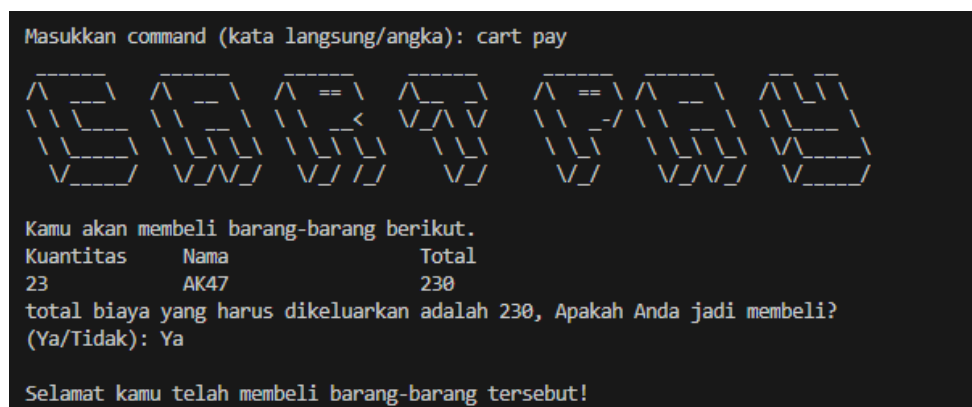
Gambar 5.8.1 Keranjang berisi barang



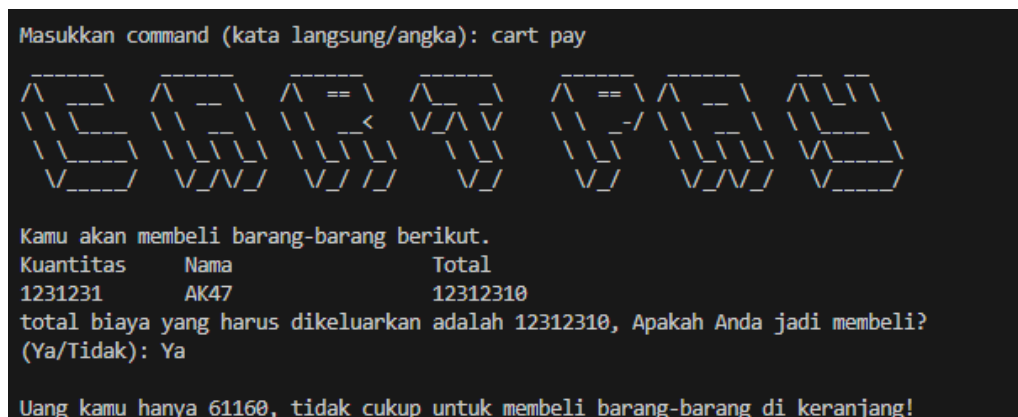
Gambar 5.8.2 Keranjang kosong

5.9 Data Test Cart Pay

Test ini memastikan program dapat membeli barang yang ada dalam keranjang dengan ketentuan pengguna memiliki uang yang cukup untuk membeli seluruh barang. Test ini juga memastikan uang yang dimiliki pengguna berkurang dan terdapat tambahan pada riwayat pembelian.



Gambar 5.9.1 Barang-barang di keranjang berhasil dibeli



Gambar 5.9.2 Barang-barang di keranjang melebihi uang yang dimiliki


```
Masukkan command (kata langsung/angka): cart pay

      /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\
     /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
    /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
   /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
  /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
 /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
/  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
\  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
 \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
  \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
   \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
    \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
     \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
      \__/\  \__/\  \__/\  \__/\  \__/\  \__/\  \__/\

Kamu akan membeli barang-barang berikut.
Kuantitas      Nama      Total
2              AK47      20
total biaya yang harus dikeluarkan adalah 20, Apakah Anda jadi membeli?
(Ya/Tidak): Tidak
```

Gambar 5.9.3 Barang-barang di keranjang tidak jadi dibeli

```
Masukkan command (kata langsung/angka): cart pay

      /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\
     /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
    /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
   /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
  /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
 /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
/  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
\  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
 \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
  \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
   \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
    \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
     \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
      \__/\  \__/\  \__/\  \__/\  \__/\  \__/\  \__/\

Kamu akan membeli barang-barang berikut.
Kuantitas      Nama      Total
2              AK47      20
total biaya yang harus dikeluarkan adalah 20, Apakah Anda jadi membeli?
(Ya/Tidak): skibidi
```

Gambar 5.9.4 Input tidak sesuai

5.10 Data Test History

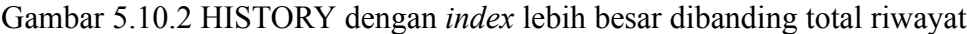
Test ini dilakukan untuk memastikan program dapat menunjukkan riwayat pembelian dari pengguna.

```
Masukkan command (kata langsung/angka): HISTORY 1

      /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\
     /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
    /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
   /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
  /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
 /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
/  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
\  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
 \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
  \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
   \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
    \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
     \  _  / \  _  / \  _  / \  _  / \  _  / \  _  /
      \__/\  \__/\  \__/\  \__/\  \__/\  \__/\  \__/\

Riwayat pembelian barang:
1. AK47 dengan harga 230
```

Gambar 5.10.1 HISTORY dengan *index* lebih kecil dibanding total riwayat



5.11 Data Test Wishlist Add

Test ini dilakukan untuk menambahkan barang ke *wishlist*.



STEI- ITB	IF2111-TB-02-12	Halaman 18 dari 42 halaman
<p>Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.</p>		



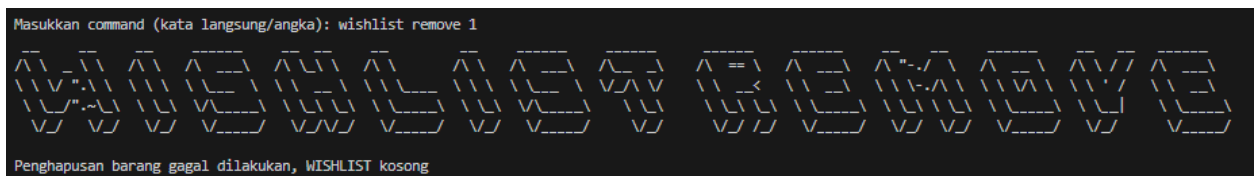
Gambar 5.13.1 Penghapusan barang gagal dilakukan



Gambar 5.13.2 Penghapusan barang barang posisi ke-i berhasil



Gambar 5.13.3 Penghapusan barang gagal karena $i > N$



Gambar 5.13.4 Penghapusan barang gagal karena *wishlist* kosong

5.14 Data Test Wishlist Remove

Test ini memastikan penghapusan barang dari *wishlist* berdasarkan input nama dari pengguna.



Gambar 5.14.1 Penghapusan barang berhasil

```
Masukkan command (kata langsung/angka): wishlist remove

Nama barang yang ingin dihapus dari WISHLIST: hahaiii
Penghapusan barang WISHLIST gagal dilakukan, hahaiii tidak ada di WISHLIST!
```

Gambar 5.14.2 Penghapusan barang gagal karena nama tidak ada

5.15 Data Test Wishlist Clear

Test ini dilakukan untuk menghapus semua barang di *wishlist*.

```
Masukkan command (kata langsung/angka): wishlist clear

WISHLIST telah dikosongkan
```

Gambar 5.15 Menghapus seluruh barang di *wishlist*

5.16 Data Test Wishlist Show

Test ini menunjukkan barang yang telah dimasukkan ke *wishlist*.

```
Masukkan command (kata langsung/angka): wishlist show

Berikut adalah isi wishlist:
1. Meong
2. AK47
Tekan karakter apapun untuk melanjutkan:
```

Gambar 5.16.1 Menampilkan barang yang ada di *wishlist*

```
Masukkan command (kata langsung/angka): wishlist show

WISHLIST kosong
```

Gambar 5.16.2 Menampilkan *wishlist* kosong

6 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Fitur Start	Memastikan sistem dapat membaca <i>file</i>	Melakukan <i>compile file</i> “main.c” di terminal	Data Test 1	Sistem berhasil membaca <i>file</i>	Hasil sesuai yang diharapkan pada Gambar 5.1
2	Fitur Load	Memastikan sistem dapat me-load <i>file</i> yang telah tersimpan	Memasukkan <i>command</i> LOAD <namafile.txt>	Data Test 2	Sistem berhasil me-load <i>file</i>	Hasil sesuai yang diharapkan pada Gambar 5.2
3.	Fitur Save	Memastikan sistem menyimpan <i>state</i> aplikasi terbaru	Mengetik <i>command</i> SAVE <namafile.txt>	Data Test 3	Sistem berhasil menyimpan <i>state</i> aplikasi terbaru	Hasil sesuai yang diharapkan pada Gambar 5.3
4.	Fitur Store List	Memastikan sistem menampilkan barang yang ada di toko	Mengetik <i>command</i> STORE LIST	Data Test 4	Sistem berhasil menampilkan barang yang ada di toko	Hasil sesuai yang diharapkan pada Gambar 5.4
5.	Fitur Profile	Memastikan sistem menampilkan data diri pengguna	Mengetik <i>command</i> PROFILE	Data Test 5	Sistem berhasil menampilkan data diri pengguna	Hasil sesuai yang diharapkan pada Gambar 5.5
6.	Fitur Cart Add	Memastikan sistem dapat menambahk	Mengetik <i>command</i> CART ADD <nama><n>	Data Test 6	Sistem berhasil menambahkan	Hasil sesuai yang diharapkan

		an barang dengan kuantitas N			barang dengan kuantitas n	pada Gambar 5.6.1
7.	Fitur <i>Cart Add</i>	Memastikan sistem memberikan pesan jika barang yang ingin ditambahkan tidak ada	Menetik <i>command</i> CART ADD <nama><n>	Data Test 6	Sistem berhasil memberikan pesan jika barang tidak ada di toko	Hasil sesuai yang diharapkan pada Gambar 5.6.2
8.	Fitur <i>Cart Remove</i> <nama><n>	Memastikan sistem dapat mengurangi barang dengan kuantitas N	Menetik <i>command</i> CAR REMOVE <nama><n>	Data Test 7	Sistem berhasil mengurangi barang dengan kuantitas n	Hasil sesuai yang diharapkan pada Gambar 5.7.1
9.	Fitur <i>Cart Remove</i> <nama><n>	Memastikan sistem memberikan pesan jika jumlah barang yang ada kurang dari jumlah barang yang ingin dikurangi	Menetik <i>command</i> CAR REMOVE <nama><n>	Data Test 7	Sistem berhasil memberi pesan tidak berhasil karena barang yang berada di keranjang lebih sedikit dari barang yang ingin dikurangi	Hasil sesuai yang diharapkan pada Gambar 5.7.2
10.	Fitur <i>Cart Remove</i> <nama><n>	Memastikan sistem memberi pesan jika barang yang ingin	Menetik <i>command</i> CAR REMOVE <nama><n>	Data Test 7	Sistem berhasil memberi pesan jika barang yang ingin	Hasil sesuai yang diharapkan pada Gambar 5.7.3

		dihapus tidak ada di keranjang			dikurangi tidak ada di keranjang	
11.	Fitur <i>Cart show</i>	Memastikan sistem menampilkan barang yang ada di <i>cart</i> dan juga menampilkan biaya yang harus dikeluarkan untuk membeli semua yang ada di keranjang	Menetik <i>command</i> CART SHOW	Data Test 8	Sistem berhasil menampilkan barang yang ada di <i>cart</i> dan total biaya yang harus dikeluarkan untuk membeli semua barang yang ada di keranjang	Hasil sesuai yang diharapkan pada Gambar 5.8.1
12.	Fitur <i>Cart show</i>	Memastikan sistem menampilkan barang yang ada di <i>cart</i>	Menetik <i>command</i> CART SHOW	Data Test 8	Sistem berhasil menampilkan pesan jika tidak ada barang di <i>cart</i>	Hasil sesuai yang diharapkan pada Gambar 5.8.2
13.	Fitur <i>Cart pay</i>	Memastikan sistem dapat membeli barang yang ada di <i>cart</i>	Menetik <i>command</i> CART PAY dan <i>command</i> Ya	Data Test 9	Sistem berhasil dapat membeli barang yang ada di <i>cart</i>	Hasil sesuai yang diharapkan pada Gambar 5.9.1
14.	Fitur <i>Cart pay</i>	Memastikan sistem memberi pesan jika uang kita	Menetik <i>command</i> CART PAY dan <i>command</i> Ya	Data Test 9	Sistem berhasil memberi pesan jika uang tidak	Hasil sesuai yang diharapkan pada Gambar 5.9.2

		kurang untuk membeli keranjang			cukup untuk membeli keranjang	
15.	Fitur <i>Cart pay</i>	Memastikan sistem kembali ke main menu ketika kita tidak ingin membeli barang	Menetik <i>command</i> CART PAY dan <i>command</i> Tidak	Data Test 9	Sistem berhasil kembali ke <i>main menu</i>	Hasil sesuai yang diharapkan pada Gambar 5.9.3
16.	Fitur <i>Cart pay</i>	Memastikan sistem memberi pesan ketika input yang dimasukkan aneh	Menetik <i>command</i> CART PAY dan <i>command</i> Tidak	Data Test 9	Sistem berhasil kembali ke <i>main menu</i>	Hasil sesuai yang diharapkan pada Gambar 5.9.4
17.	Fitur <i>History</i>	Memastikan sistem menampilkan riwayat pembelian pengguna dengan total yang diinginkan	Menetik <i>command</i> HISTORY<n>	Data Test 10	Sistem berhasil menampilkan riwayat pembelian pengguna sebanyak yang diinginkan	Hasil sesuai yang diharapkan pada Gambar 5.10.1
18.	Fitur <i>History</i>	Memastikan sistem menampilkan riwayat pembelian pengguna dengan total	Menetik <i>command</i> HISTORY<n>	Data Test 10	Sistem berhasil menampilkan riwayat pembelian pengguna	Hasil sesuai yang diharapkan pada Gambar 5.10.2

		yang diinginkan			sebanyak yang diinginkan	
19.	Fitur <i>History</i>	Memastikan sistem memberi pesan jika user tidak pernah membeli barang apapun	Menetik <i>command</i> HISTORY<n>	Data Test 10	Sistem berhasil memberi pesan jika user belum memberi barang	Hasil sesuai yang diharapkan pada Gambar 5.10.3
20.	Fitur <i>Wishlist Add</i>	Memastikan sistem dapat menambahkan barang ke <i>wishlist</i>	Menetik <i>command</i> WISHLIST ADD dan nama barang yang ingin dimasukkan	Data Test 11	Sistem berhasil menambahkan barang ke <i>wishlist</i>	Hasil sesuai yang diharapkan pada Gambar 5.11.3
21.	Fitur <i>Wishlist Add</i>	Memastikan sistem memberi pesan jika barang yang ingin dimasukkan sudah ada di <i>wishlist</i>	Menetik <i>command</i> WISHLIST ADD dan nama barang yang ingin dimasukkan	Data Test 11	Sistem berhasil memberi pesan jika barang sudah ada di <i>wishlist</i>	Hasil sesuai yang diharapkan pada Gambar 5.11.1
22	Fitur <i>Wishlist Add</i>	Memastikan sistem memberi pesan jika barang yang ingin dimasukkan tidak ada di <i>store</i>	Menetik <i>command</i> WISHLIST ADD dan nama barang yang ingin dimasukkan	Data Test 11	Sistem berhasil memberi pesan jika barang tidak ada di <i>store</i>	Hasil sesuai yang diharapkan pada Gambar 5.11.2

23.	Fitur <i>Wishlist Swap</i> <i><j>	Memastikan sistem dapat menukar barang dari posisi ke-i dengan barang posisi ke-j	Menetik <i>command</i> WISHLIST SWAP <i><j>	Data Test 12	Sistem berhasil menukar barang di <i>wishlist</i>	Hasil sesuai yang diharapkan pada Gambar 5.12.1
24	Fitur <i>Wishlist Swap</i> <i><j>	Memastikan sistem memberi pesan gagal ketika hanya terdapat satu barang di <i>wishlist</i>	Menetik <i>command</i> WISHLIST SWAP <i><j>	Data Test 12	Sistem berhasil memberi pesan bahwa gagal menukar posisi <i>wishlist</i>	Hasil sesuai yang diharapkan pada Gambar 5.12.2
25.	Fitur <i>Wishlist Remove</i> <i>	Memastikan sistem dapat menghapus barang dengan posisi ke-i dari <i>wishlist</i>	Menetik <i>command</i> WISHLIST REMOVE	Data Test 13	Sistem berhasil menghapus barang dari <i>wishlist</i>	Hasil sesuai yang diharapkan pada Gambar 5.13.2
26.	Fitur <i>Wishlist Remove</i> <i>	Memastikan sistem memberi pesan jika barang ke-i tidak ada di <i>wishlist</i>	Menetik <i>command</i> WISHLIST REMOVE	Data Test 13	Sistem berhasil memberi pesan gagal karena barang ke-i tidak ada	Hasil sesuai yang diharapkan pada Gambar 5.13.3
27.	Fitur <i>Wishlist Remove</i> <i>	Memastikan sistem memberi pesan gagal karena	Menetik <i>command</i> WISHLIST REMOVE	Data Test 13	Sistem berhasil memberi pesan gagal	Hasil sesuai yang diharapkan pada Gambar 5.13.4

		<i>wishlist</i> kosong			karena <i>wishlist</i> kosong	
28.	Fitur <i>Wishlist</i> <i>Remove</i> <i>	Memastikan sistem memberi pesan karena <i>command</i> tidak <i>valid</i>	Menetik <i>command</i> WISHLIST REMOVE	Data Test 13	Sistem berhasil memberi pesan karena <i>command</i> tidak <i>valid</i>	Hasil sesuai yang diharapkan pada Gambar 5.13.1
30.	Fitur <i>Remove</i>	Memastikan sistem dapat menghapus berdasarkan nama yang dimasukkan pengguna	Menetik <i>command</i> WISHLIST REMOVE	Data Test 14	Sistem berhasil menghapus barang dari <i>wishlist</i>	Hasil sesuai yang diharapkan pada Gambar 5.14.1
31.	Fitur <i>Remove</i>	Memastikan sistem memberi pesan karena nama yang ingin dihapus tidak ada di <i>wishlist</i>	Menetik <i>command</i> WISHLIST REMOVE	Data Test 14	Sistem berhasil memberi pesan ketika nama tidak ada di <i>wishlist</i>	Hasil sesuai yang diharapkan pada Gambar 5.14.2
32.	Fitur <i>Wishlist</i> <i>Clear</i>	Memastikan sistem dapat menghapus semua barang di <i>wishlist</i>	Menetik <i>command</i> WISHLIST CLEAR	Data Test 15	Sistem berhasil menghapus semua barang dari <i>wishlist</i>	Hasil sesuai yang diharapkan pada Gambar 5.15
33.	Fitur <i>Wishlist</i> <i>Show</i>	Memastikan sistem dapat menampilkan barang	Menetik <i>command</i> WISHLIST SHOW	Data Test 16	Sistem berhasil menampilkan	Hasil sesuai yang diharapkan

		yang ada di <i>wishlist</i>			barang yang ada di <i>wishlist</i>	pada Gambar 5.16.1
34.	Fitur <i>Wishlist Show</i>	Memastikan sistem memberi pesan ketika <i>wishlist</i> kosong	Mengetik <i>command</i> WISHLIST SHOW	Data Test 16	Sistem berhasil memberi pesan karena <i>wishlist</i> kosong	Hasil sesuai yang diharapkan pada Gambar 5.16.2

7 Pembagian Kerja dalam Kelompok

No.	Fitur/ADT	NIM Coder
1	Fitur <i>Start</i>	18223015
2	Fitur <i>Load</i>	18223027
3	Fitur <i>Save</i>	18223027
4	Fitur <i>Store List</i>	18223015
5	Fitur <i>Profile</i>	18223001
6	Fitur <i>Cart Add</i>	18223001
7	Fitur <i>Cart Remove</i>	18223001
8	Fitur <i>Cart Show</i>	18223001
9	Fitur <i>Cart Pay</i>	18223001

10	Fitur <i>History</i>	18223001
11	Fitur <i>Wishlist Add</i>	18223015
12	Fitur <i>Wishlist Swap</i>	18223015
13	Fitur <i>Wishlist Remove <i></i>	18223015
14	Fitur <i>Wishlist Remove</i>	18223015
15	Fitur <i>Wishlist Clear</i>	18221018
16	Fitur <i>Wishlist Show</i>	18221018
17	ADT	18223015
18	Main	18223015
19	Laporan	18221018, 18223001, 18223011, 18223015, 18223027, 18223091

8 Lampiran

8.1 Deskripsi Tugas Besar

■ 8.1.1 Latar Belakang

Agen Purry sedang menikmati tidur siangya ketika dia tiba-tiba mendengar alarm dari belakang sofa. Suatu pintu rahasia terbuka di bawah dirinya dan ia jatuh ke ruang bawah tanah dan langsung disambut dengan misi terbarunya.

“Ah Agen Purry, maaf harus mengganggu waktu tidur kamu tapi kami mendapatkan laporan bahwa Dr. Asep Spakbor sedang membuat suatu mesin yang dinamakan ‘Oppenheimer-inator’

yang akan menghancurkan wilayah tiga negara bagian. Aku membutuhkan bantuanmu untuk menghentikan Dr. Asep Spakbor, ini merupakan ancaman terbesar yang pernah ia buat.”

And indeed it was. Setelah pertarungan sengit selama 3 bulan 13 hari 2 jam 47 menit dan 2 detik, suplai senjata dan suplai peralatan yang dimiliki OWCA mulai menipis. Harapan kemenangan OWCA mulai memudar...

Tanpa disangka, Agen Purry mengeluarkan senjata rahasia miliknya: menjadi orang Bojongsoang yang memiliki kenalan pegawai Borma. Toko Borma adalah komponen penting yang dapat membawakan kemenangan untuk OWCA pada waktu-waktu kritis ini. Sebab, meskipun Borma terlihat seperti supermarket pada umumnya, mereka sebenarnya merupakan pemasok barang-barang perang. Namun terdapat satu masalah kecil, Borma masih beroperasi secara tatap muka dan OWCA tidak memiliki transport untuk pergi ke Bojongsoang.

Untuk menyelesaikan permasalahan ini, OWCA mengontak tim programmer paling andalnya untuk merancang suatu sistem jual beli ke Borma dengan nama PURRMART! Benar, tim tersebut adalah kalian! Misi ini akan menantang dan menguji kalian. Namun, dengan kerja tim dan tekad yang kuat, kalian pasti dapat menghadapi tantangan ini.

■ 8.1.2 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.

■ 8.1.3 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. PROFILE

PROFILE adalah command yang digunakan untuk melihat data diri pengguna. PROFILE hanya dapat dipanggil saat status pengguna telah login

B. CART ADD <nama> <n>

CART ADD adalah command yang digunakan untuk menambahkan barang dengan kuantitas tertentu ke dalam keranjang belanja.

C. CART REMOVE <nama> <n>

CART REMOVE adalah command yang digunakan untuk 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.

D. CART SHOW

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

E. CART PAY

CART PAY adalah command yang digunakan untuk 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.

Nama barang yang dimasukan ke riwayat pembelian adalah barang dengan total harga (harga barang * kuantitas) terbesar. Jika terdapat lebih dari 1 barang dengan total yang sama, maka yang disimpan adalah barang dengan urutan lexical yang lebih besar.

Dimasukan juga total harga pada pembelian tersebut.

F. HISTORY<n>

HISTORY adalah command yang digunakan untuk 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.

G. WISHLIST ADD

WISHLIST ADD merupakan command yang digunakan untuk menambahkan suatu barang ke wishlist.

H. WISHLIST SWAP <i> <j>

WISHLIST SWAP merupakan command yang digunakan untuk 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.

I. WISHLIST REMOVE <i>

WISHLIST REMOVE adalah command yang digunakan untuk menghapus barang dengan posisi ke-i dari wishlist.

J. WISHLIST REMOVE

WISHLIST REMOVE adalah command yang digunakan untuk menghapus barang dari wishlist berdasarkan nama barang yang dimasukkan pengguna.

K. WISHLIST CLEAR

WISHLIST CLEAR adalah command yang digunakan untuk menghapus semua barang yang terdapat di dalam WISHLIST.

PERUBAHAN COMMAND

Terdapat beberapa command iterasi sebelumnya yang berubah, yaitu sebagai berikut.

a. START, LOAD, dan SAVE

Terdapat perubahan konfigurasi yang harus ditambahkan dalam implementasi command START, LOAD, dan SAVE.

b. STORE LIST

STORE LIST akan menampilkan nama barang yang dijual beserta harganya.

■ **8.1.4 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:

STEI- ITB	IF2111-TB-02-12	Halaman 34 dari 42 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

1. 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>**
 - c. Baris selanjutnya adalah bilangan bulat positif **J** yang menunjukkan banyaknya *wishlist* pengguna tersebut
 - d. Selanjutnya, sejumlah **J** baris menyatakan nama barang dengan format **<Nama barang>**

■ 8.1.5 Daftar ADT

1. ADT Kustom

Terdapat perubahan pada ADT pengguna (User). ADT akan menyimpan keranjang, riwayat pembelian, dan wishlist yang dimiliki oleh seorang user.

2. ADT Stack

ADT ini digunakan untuk menyimpan riwayat pembelian seorang pengguna. Riwayat pembelian akan menyimpan nama barang dengan ketentuan di bawah dan total biaya seluruh barang.

Nama barang yang disimpan adalah barang dengan total harga (harga barang * kuantitas) terbesar. Jika terdapat lebih dari 1 barang dengan total yang sama, maka yang disimpan

adalah barang dengan urutan lexical yang lebih besar. Total biaya dari seluruh pembelian juga disimpan pada suatu elemen

3. ADT Setmap

ADT ini digunakan untuk 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.

4. ADT Linked List

ADT ini digunakan untuk menyimpan wishlist seorang pengguna. Elemen yang disimpan di dalam wishlist adalah nama barang yang ingin dibeli.

■ 8.1.6 Bonus

Pada tugas besar ini, terdapat beberapa fitur tambahan yang bisa diimplementasikan. Fitur-fitur ini tidak wajib untuk dikerjakan dan bobotnya lebih kecil dibanding fitur utama. **Utamakan fitur-fitur utama yang diminta sebelum mengerjakan bonus.** Berikut adalah penjelasan dari masing-masing fitur bonus:

1. 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.

2. Riwayat Maksimal

ADT Stack digunakan untuk menyimpan riwayat pembelian seorang pengguna. Namun, data yang disimpan hanya nama barang dengan total harga terbesar. Petinggi OWCA ingin memiliki detail setiap pembelian ~~untuk keperluan *tax fraud*~~, mereka ingin *command*

HISTORY untuk menunjukkan seluruh barang yang dibeli.

Agar detail riwayat pembelian *persistent*, terdapat juga perubahan pada konfigurasi file, yaitu:

- a. Tidak terdapat **K** baris riwayat, tetapi elemen riwayat.
- b. Baris pertama setiap elemen riwayat adalah bilangan positif tidak nol **L** yang menunjukkan banyaknya barang yang dibeli dalam pembelian dan bilangan **X** yaitu total harga pada pembelian tersebut
- c. Selanjutnya, sejumlah **L** baris menyatakan barang yang dibeli dengan format **<Total biaya> <Kuantitas barang> <Nama barang>**

3. Deteksi Kebocoran Senjata Biologis

Pada *milestone* sebelumnya, beberapa dari Anda telah sukses membuat sistem untuk mendeteksi kode pada DNA dari pabrik untuk mencegah sabotase musuh. Namun, OWCA mencurigai adanya kebocoran pada gudang PURRMART akibat penyimpanan yang tidak sesuai prosedur operasional baku (POB). Untuk menyelidiki hal tersebut, OWCA mencoba melakukan metagenomik untuk mengetahui spesies yang terdapat pada sampel dari lingkungan. Proses tersebut membutuhkan proses [sequence alignment](#) untuk mengecek kesamaan antara sekuens senjata biologis dengan sekuens hasil metagenomik. Oleh karena itu, Anda diminta mengimplementasikan kaskas *global alignment* dengan algoritma [Needleman-Wunsch](#). Apabila skor akhir lebih besar dari 80% panjang sekuens yang lebih panjang (jumlah karakter/basa nukleotida), maka dapat disimpulkan bahwa terjadi kebocoran senjata biologis. Panjang sekuens maksimum 50 karakter. Panjang kedua sekuens tidak harus sama, tetapi cukup mirip (misal 48 karakter dan 44 karakter). **Perhatikan kompleksitas algoritma, tidak boleh lebih dari $\sim O(2^n)$.**

Ketentuan *scoring*:

- *Match*: +1
- *Mismatch*: 0

- *Gap penalty*: -1

4. Optimasi Rute Ekspedisi

Toko PURRMART memiliki banyak klien sehingga perusahaan SiLambat, rekan toko PURRMART, membutuhkan cara untuk mengirim barang dengan rute yang paling efisien. Seluruh titik harus dikunjungi dan suatu titik ke titik lainnya memiliki jarak tertentu. Awalnya, salah satu karyawan PURRMART, menggunakan algoritma BFS untuk menyelesaikan permasalahan ini. Namun, ternyata algoritma tersebut tidak efisien dan memerlukan kemampuan komputasi yang besar. Anda diminta menggunakan algoritma alternatif yang lebih efisien untuk menyelesaikan masalah ini, semakin efisien semakin baik. **Jelaskan alasan pemilihan algoritmanya di laporan.**

5. Pencarian Barang

Toko PURRMART memiliki terlalu banyak barang sehingga kamu diminta untuk membantu membuat sebuah mesin pencari baru. Mesin pencari yang lama tidak efektif dikarenakan mesin tersebut mencari barang yang memiliki nama secara *exact match*. Karena manusia tempatnya salah dan lupa, barang yang dicari sering tidak ketemu karena nama barang yang dicari seringlah sekelumit berbeda. Oleh karena itu, kamu diminta untuk membuat mesin pencari kuasi match menggunakan jarak Levenshtein dengan *threshold distance default* adalah 10. *Threshold distance default* dapat diganti.

■ 8.1.7 Catatan Tambahan

1. Tampilan program boleh dibuat sesuai keinginan kalian, tampilan pada spesifikasi ini hanya merupakan contoh.
2. **Diwajibkan** untuk membuat **driver** untuk masing-masing ADT. *Driver* berisi sebuah *main file* yang memanggil fungsi/prosedur yang ada di ADT tersebut. Kegunaan *driver* adalah untuk *testing* ADT yang sudah dibuat.
3. Sebagai saran, manfaatkan **Makefile** untuk mempermudah proses kompilasi dan penjalanan program. Bila sulit dalam menggunakan **Makefile**, bisa diakali dengan menggunakan **shell script/batch file**.

4. Gunakan **GitHub** sebagai *version control*, lalu undang asisten kalian sebagai *collaborator*. Pastikan asisten sudah masuk ke dalam *repository* sebelum asistensi pertama.
5. Buat *file* README yang minimal mengandung deskripsi singkat program, identitas anggota kelompok dan cara kompilasi program. *Readme* dapat
6. dibuat dengan menggunakan [markdown](#).
7. Buat struktur program yang serapi mungkin. Jangan buat semuanya pada *file* yang sama. Contoh struktur program (tidak harus diikuti):
8. Manfaatkan ADT yang sudah kalian buat dalam praktikum semaksimal mungkin.
9. Perhatikan bahwa nilai untuk bonus akan **lebih kecil** dibandingkan dengan fitur utama. Silakan prioritaskan fitur-fitur yang lebih penting terlebih dahulu.
10. Jika ada yang kurang jelas, silahkan bertanya melalui [QnA](#).




8.2 Notulen Rapat





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

No. Kelompok/Kelas : 12/01
Nama Kelompok : GOKS
Anggota Kelompok (Nama/NIM) :
1. Jennifer Grachel / 18221018
2. Darren Mansyl / 18223001
3. Samuel Chris Michael Bagasta Simanjuntak / 18223011
4. Muhammad Adam Mirza / 18223015
5. Ferro Arka Berlian / 18223027
6. Jason Samuel / 18223091

Asisten Pembimbing : Rayhan Maheswara Pramanda

Asistensi I

Tanggal : 18 Desember 2024	Catatan Asistensi: <ol style="list-style-type: none"> 1. Inputan harus satu baris bagi fungsi yang tertera di spesifikasi. 2. Untuk bonus Store List Gacor harus mengganti fungsi store-store lainnya.
Tempat : Google Meet	
Kehadiran Anggota Kelompok: <p>No</p> <p>NIM</p> <p>Tanda tangan</p> <p>1</p> <p>18223011</p>  <p>2</p> <p>18223001</p>  <p>3</p> <p>18223027</p>  <p>4</p> <p>18223015</p>	

 5 18223091  6 18221018 	
	Tanda Tangan Asisten: 

8.3 Log Activity Anggota Kelompok

No.	Tanggal	Keterangan
1	December 9 2024	Pembagian tugas

2	10-18 December 2024	Pengerjaan tugas masing-masing
3	18 December 2024	Asistensi II
4	20-22 December 2024	Pengerjaan bersama dan finalisasi