

LAPORAN TUGAS BESAR


IF2111 Algoritma dan Struktur Data STI

PURRMART

Dipersiapkan oleh:
Kelompok 1

Seren Elizabeth Siahaan	18221160
Zaka Hanif Nabalah	18223006
Rayhan Hidayatul Fikri	18223022
Favian Rafi Laftiyanto	18223036
Wijasara Aptaluhung	18223088

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

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		IF2111-TB-02-01		54
		Revisi	0	20 Desember 2024

Daftar Isi

1	Ringkasan.....	6
2	Penjelasan Tambahan Spesifikasi Tugas.....	8
	2.1 GLOBALALIGNMENT.....	8
	2.2 OPTIMASIRUTE.....	8
3	Struktur Data (ADT).....	11
	3.1 User.....	11
	3.2 Array.....	11
	3.3 Stack.....	11
	3.4 Setmap.....	12
	3.5 Linked List.....	13
4	Program Utama.....	14
5	Algoritma-Algoritma Menarik.....	18
	5.1 Needleman-Wunsch.....	18
	5.2 Backtracking.....	19
	5.3 keepLastWord dan removeLastWord.....	20
6	Data Test.....	20
	6.1 Data Test Profile.....	21
	6.2 Data Test Cart Add <nama> <n>.....	21
	6.3 Data Test Cart Remove <nama> <n>.....	22
	6.4 Data Test Cart Show.....	22
	6.5 Data Test Cart Pay.....	23
	6.6 Data Test History <n>.....	24
	6.7 Data Test Wishlist Add.....	25
	6.8 Data Test Wishlist Swap <i> <j>.....	26
	6.9 Data Test Wishlist Remove <i>.....	27
	6.10 Data Test Wishlist Remove.....	27
	6.11 Data Test Wishlist Clear.....	27
	6.12 Data Test Wishlist Show.....	28
	6.13 Data Test Perubahan Start, Load, dan Save.....	28
	6.14 Data Test Perubahan Store List.....	31
	6.15 Data Test Bonus Globalalignment (Deteksi Kebocoran Senjata Biologis)....	

6.16 Data Test Bonus Optimasirute.....	32
7 Test Script.....	33
8 Pembagian Kerja dalam Kelompok.....	40
9 Lampiran.....	42
9.1 Deskripsi Tugas Besar.....	42
Command.....	42
user dapat memasukkan command-command berikut.....	42
a. PROFILE.....	42
b. CART ADD <nama> <n>.....	42
c. CART REMOVE <nama> <n>.....	43
d. CART SHOW.....	43
e. CART PAY.....	43
f. HISTORY <n>.....	44
g. WISHLIST ADD.....	44
h. WISHLIST SWAP <i> <j>.....	44
i. WISHLIST REMOVE <i>.....	44
j. WISHLIST REMOVE.....	44
k. WISHLIST CLEAR.....	44
l. WISHLIST SHOW.....	45
m. Perubahan Command START, LOAD, dan SAVE.....	45
n. Perubahan Command STORE LIST.....	45
o. Deteksi Kebocoran Senjata Biologis.....	45
p. Optimasi Rute Ekspedisi.....	46
9.2 Notulen Rapat.....	46
9.3 Log Activity Anggota Kelompok.....	49

Daftar Gambar

Gambar 1. Tampilan Profile ketika dijalankan.....	21
Gambar 2. Tampilan Cart Add ketika berhasil dijalankan.....	21
Gambar 3. Tampilan Cart Add ketika barang tidak ada di toko.....	21
Gambar 4. Tampilan Cart Add ketika inputan tidak sesuai ketentuan.....	22
Gambar 5. Tampilan Cart Remove <nama> <n> ketika berhasil dijalankan.....	22
Gambar 6. Tampilan Cart Remove <nama> <n> ketika jumlah barang tidak memenuhi.....	22
Gambar 7. Tampilan Cart Remove <nama> <n> ketika inputan tidak sesuai ketentuan.....	22
Gambar 8. Tampilan Cart Show ketika dijalankan dan menampilkan semua cart..	23
Gambar 9. Tampilan Cart Show ketika isi cart kosong.....	23
Gambar 10. Tampilan Cart Pay ketika menampilkan semua barang.....	24
Gambar 11. Tampilan Cart Pay ketika ingin membeli barang.....	24
Gambar 12. Tampilan Cart Pay ketika barang dibatalkan pembelian.....	24
Gambar 13. Tampilan Cart Pay ketika input tidak valid.....	24
Gambar 14. Tampilan History ketika dijalankan dan sesuai inputan masih memuat.....	25
Gambar 15. Tampilan History ketika inputan lebih besar dari jumlah barang di keranjang.....	25
Gambar 16. Tampilan History ketika inputan tidak sesuai.....	25
Gambar 17. Tampilan Wishlist Add ketika dijalankan.....	26
Gambar 18. Tampilan Wishlist Add ketika tidak ada barang yang sesuai.....	26
Gambar 19. Tampilan Wishlist Add ketika barang sudah ada di wishlist.....	26
Gambar 20. Tampilan Wishlist Swap ketika dijalankan.....	26
Gambar 21. Tampilan Wishlist Swap ketika barang tidak dapat ditukar.....	27
Gambar 22. Tampilan Wishlist Swap ketika barang tidak ditemukan.....	27
Gambar 23. Tampilan Wishlist Remove <i> ketika dijalankan.....	27
Gambar 24. Tampilan Wishlist Remove <i> ketika barang yang diinginkan tidak ada.....	27
Gambar 25. Tampilan Wishlist Remove ketika dijalankan.....	27
Gambar 26. Tampilan Wishlist Remove ketika gagal dijalankan.....	27

Gambar 27. Tampilan Wishlist Remove ketika dijalankan.....	28
Gambar 28. Tampilan Wishlist Show ketika dijalankan.....	28
Gambar 29. Tampilan Wishlist Remove ketika tidak ada barang wishlist.....	28
Gambar 30. Tampilan START ketika dijalankan.....	29
Gambar 31. Tampilan LOAD <filename.txt> ketika berhasil dijalankan.....	29
Gambar 32. Tampilan LOAD <filename.txt> ketika gagal dijalankan.....	30
Gambar 33. Tampilan SAVE <filename.txt> ketika berhasil menyimpan file test.txt.....	30
Gambar 34. Tampilan SAVE <filename.txt> ketika gagal menyimpan file salah format.....	30
Gambar 35. Tampilan STORE LIST ketika berhasil dijalankan.....	31
Gambar 36. Tampilan GLOBALALIGNMENT ketika tidak ada kebocoran pada sekuens.....	31
Gambar 37. Tampilan GLOBALALIGNMENT ketika ada kebocoran pada sekuens. 32	
Gambar 38. Tampilan OPTIMASIRUTE ketika berhasil dijalankan.....	32

1 Ringkasan

PURRMART merupakan program simulasi e-commerce yang memungkinkan user untuk melakukan transaksi pembelian barang. Untuk menggunakan program, user perlu mengakses akun lamanya atau membuat akun baru. Setelah mengakses akun, user dapat melihat daftar barang, membuat permintaan masuknya barang baru atau tambahan stok barang yang ada, menampilkan riwayat pembelian barang, serta membuat dan menghapus daftar barang yang diinginkan. user dapat memanfaatkan uang yang didapatkannya untuk membeli barang. Hal ini dapat dilakukan user dengan bekerja yang kemudian user akan mendapatkan uang.

Program PURRMART menggunakan bahasa C dan sejumlah Abstract Data Type (ADT) sebagai struktur data yang diproses. Terdapat 6 ADT yang digunakan, yakni ADT User, ADT Barang, ADT List, ADT Mesin Karakter, ADT Mesin Kata, ADT Queue. ADT User dan ADT Barang digunakan untuk menyimpan informasi barang di toko atau di daftar permintaan serta menyimpan informasi akun user yang menggunakan program. ADT List diimplementasikan menggunakan struktur data array statis dan dinamis. ADT List digunakan untuk menyimpan struktur data barang dan user. ADT Mesin Karakter dan Mesin Kata digunakan untuk memproses masukan user dan pembacaan file dengan struktur data Word. ADT Queue digunakan untuk memproses daftar permintaan barang.

Ketika program PURRMART dijalankan, user akan diberikan pilihan untuk melakukan START, LOAD, atau HELP. Masukan user yang diterima hanyalah salah satu dari pilihan tersebut. Apabila masukan user tidak diterima, akan muncul permintaan untuk memasukkan kembali perintah yang valid. Setelah menjalankan salah satu perintah, user akan memulai sesi dan diberikan pilihan untuk menjalankan perintah REGISTER, LOGIN, atau QUIT. Cara kerja perintah - perintah ini telah dijelaskan dalam laporan Tugas Besar 1.

Setelah user melakukan LOGIN, user dapat menjalankan perintah - perintah yang telah kami jelaskan di laporan Tugas Besar 1, seperti WORK, WORK CHALLENGE, STORE LIST, STORE REQUEST, STORE SUPPLY, STORE REMOVE, LOGOUT, SAVE, dan QUIT. Namun, ada 15 perintah baru yang bisa dijalankan, yakni PROFILE, CART ADD <nama

barang> <jumlah barang>, CART REMOVE <nama barang> <jumlah barang>, CART SHOW, CART PAY, HISTORY, WISHLIST ADD, WISHLIST SWAP <i> <j>, WISHLIST REMOVE <i>, WISHLIST REMOVE, WISHLIST CLEAR, WISHLIST SHOW, HISTORY <n>, GLOBALALIGNMENT, dan OPTIMASIRUTE.

Perintah - perintah yang memiliki kata CART dapat digunakan user untuk memodifikasi keranjang belanjanya, yakni tempat dicatatnya nama barang dan kuantitas yang ingin dibeli oleh user. User bisa menyimpan lebih dari satu barang dalam keranjangnya dan melakukan pembayaran seluruh barang dalam satu kali pembayaran.

Perintah CART ADD <nama barang> <jumlah barang> digunakan untuk menambahkan barang dengan kuantitas tertentu ke keranjang belanja. Nama barang yang dimasukkan harus ada dalam daftar barang yang dijual di toko. Perintah CART REMOVE <nama barang> <jumlah barang> digunakan untuk mengurangi kuantitas atau bahkan menghapus barang yang ada dalam keranjang belanja. Masukan jumlah barang harus lebih kecil atau sama dengan kuantitas yang sebelumnya dipesan. Perintah CART SHOW digunakan untuk menunjukkan isi dari keranjang belanja user dalam format kuantitas barang, nama barang, dan total harga yang perlu dibayarkan per barang. Perintah CART PAY digunakan untuk menampilkan seluruh barang di keranjang beserta total harga yang harus dibayarkan. Kemudian, user diberikan pilihan untuk melakukan pembayaran atau tidak. Jika pembayaran dilakukan, saldo user akan berkurang sesuai total harga yang dibayarkan. Namun, jika saldo user tidak mencukupi, maka pembayaran akan dibatalkan.

Perintah - perintah yang memiliki kata WISHLIST dapat digunakan user untuk memodifikasi daftar wishlist-nya, yakni tempat dicatatnya nama barang yang menarik minat user tetapi belum menjadi prioritas untuk dibeli dalam waktu dekat. User bisa menyimpan lebih dari satu barang dalam wishlistnya.

Perintah WISHLIST ADD dapat digunakan untuk menambahkan barang ke wishlist user. User akan diminta input berupa nama barang dengan syarat nama barang terdapat di toko. Perintah WISHLIST SHOW dapat digunakan untuk menunjukkan daftar barang pada wishlist user dengan format nomor urutan dan nama barang dengan nomor urutan paling awal adalah barang terbaru yang dimasukkan ke wishlist. Perintah WISHLIST REMOVE <i> dapat digunakan untuk menghapus suatu barang

pada nomor urutan tertentu pada wishlist dengan *i* sebagai input nomor urutannya. Adapun perintah **WISHLIST REMOVE** yang menghapus barang pada wishlist berdasarkan masukan nama barang dari user. Perintah **WISHLIST CLEAR** dapat digunakan untuk menghapus semua barang yang terdapat dalam wishlist user.

Perintah **PROFILE** dapat digunakan untuk menampilkan nama dan saldo dari user yang sedang login dalam suatu sesi. Perintah **HISTORY <n>** dapat digunakan untuk menampilkan riwayat pembelian yang dilakukan oleh user dengan *n* sebagai jumlah riwayat yang ingin ditampilkan dengan urutan dari pembelian terbaru. Riwayat pembelian dimunculkan dengan format nama barang dan total harga yang dibayarkan. Adapun perintah **GLOBALALIGNMENT** untuk memeriksa kebocoran senjata biologis berdasarkan sekuens DNA dan **OPTIMASIRUTE** untuk menemukan rute pengiriman paling efektif bagi pelanggan.

user juga dapat menyimpan kondisi sistem terkait barang dan user ke dalam sebuah save file melalui perintah **SAVE** yang telah dimodifikasi dari tugas sebelumnya atau melalui perintah **QUIT**. Kondisi sistem yang disimpan berupa jumlah barang diikuti daftar nama barang dan harganya di toko beserta jumlah user diikuti dengan daftar nama, password, saldo, riwayat pembelian, dan wishlist tiap user.

2 Penjelasan Tambahan Spesifikasi Tugas

Pada pengembangan sistem aplikasi PURRMART ini, kami melakukan penambahan - penambahan spesifikasi berupa asumsi yang kemudian akan dijelaskan dan kami rincikan. Spesifikasi tambahan yang dimaksud adalah sebagai berikut:

2.1 **GLOBALALIGNMENT**

Pada fungsi ini, kami menambahkan kondisi ketika *user* memasukkan sekuens yang tidak valid, yaitu ketika ada karakter selain 'A', 'G', 'T', dan 'C'. Pada kondisi ini, program akan mencetak tulisan:

Masukan sekuens tidak valid.

2.2 **OPTIMASIRUTE**

Pada fungsi ini, kami menambahkan asumsi bahwa banyak *node* atau lokasi pengiriman tidak lebih dari 50. Lalu, kami juga membuat asumsi bahwa graf

merupakan graf simpel sehingga jumlah *edges* maksimal dan minimal pada suatu graf dengan *node* sebanyak n berturut-turut adalah $\frac{n(n-1)}{2}$ dan $(n - 1)$. Kami lalu juga berasumsi bahwa jarak antara dua *node* atau lokasi pengiriman tidak mungkin nol atau negatif. Sejalan juga asumsi mengenai graf simpel, kami juga berasumsi bahwa tidak ada *multiple edges* sehingga jarak antar-*edges* hanya ada satu. Kami juga berasumsi bahwa tidak ada *loop* atau *edge* yang mengarah pada dua *node* yang sama. Maka dari itu, *by default* kami mengeset jarak antar-*node* pada dua *nodes* yang sama adalah nol. Kami juga membuat asumsi bahwa rute selalu mulai dari *node* 0. Karena asumsi-asumsi tersebut, kami pun menambahkan tulisan pada program jika asumsi yang kami buat dilanggar. Ketika lokasi pengiriman ada lebih dari 50, program akan mencetak:

```
Jumlah lokasi pengiriman melebihi batas maksimal. Silakan coba lagi.
```

Ketika lokasi pengiriman hanya ada 1, tidak ada rute yang perlu dibuat. Maka dari itu, program akan mencetak:

```
Wah, lokasi pengiriman hanya ada 1. Tidak ada rutenya, dong :(
```

Lalu, ketika lokasi pengiriman kurang dari 1, program akan mencetak:

```
Jumlah lokasi pengiriman tidak valid. Silakan coba lagi.
```

Saat jumlah *edges* atau rute antar-*node* kurang dari jumlah *edges* minimal, program akan mencetak:

```
Jumlah rute terlalu sedikit. Silakan coba lagi.
```

Lalu, saat jumlah *edges* atau rute antar-*node* lebih dari jumlah *edges* maksimal, program akan mencetak:

```
Jumlah rute melebihi batas. Silakan coba lagi.
```

Ketika *user* memasukkan *node* yang tidak ada (*nonexistent*) di graf, yaitu kurang dari nol dan lebih besar dari jumlah *node* dikurangi satu, saat memasukkan jarak antar-lokasi pengantaran, program akan mencetak:

```
Node yang kamu masukkan tidak valid. Silakan coba lagi.
```

Lalu, ketika *user* memasukkan dua *node* yang sama saat memasukkan jarak antar-lokasi pengantaran, program akan mencetak:

```
Kamu tidak bisa membuat jarak ke node yang sama. Silakan coba lagi.
```

Selanjutnya, ketika *user* memasukkan jarak yang kurang dari sama dengan nol, program akan mencetak:

```
Jarak yang kamu masukkan tidak valid. Silakan coba lagi.
```

Kami juga menambahkan kondisi lain, yaitu ketika *user* memasukkan jarak pada *edge* yang telah diberi jarak sebelumnya. Pada kondisi ini, program akan mencetak:

```
Anda sudah memasukkan jarak di antara kedua node ini. Silakan coba node lainnya.
```

Dalam algoritma ini sendiri, kami juga berasumsi bahwa tiap *edges* dan *nodes* hanya dapat dikunjungi sekali. Sejalan dengan ini, kami memperkirakan bahwa ada kemungkinan tidak semua *nodes* dapat dikunjungi. Pada kondisi ini, program akan mencetak:

```
Tidak semua node dapat dikunjungi tepat sekali. Silakan coba lagi.
```

Secara keseluruhan, ada kemungkinan bahwa *user* memasukkan *input* yang tidak sesuai dengan apa yang seharusnya dimasukkan, seperti contohnya *user* memasukkan *input* berupa huruf dan bukan angka. Apabila hal ini terjadi, program akan mencetak:

```
Jumlah yang kamu masukkan tidak valid. Silakan coba lagi.
```

Ketika *input* yang dimasukkan kosong, program akan mencetak:

```
Masukan kamu kosong. Silakan coba lagi.
```

3 Struktur Data (ADT)

3.1 *User*

File `user.h` digunakan untuk memodelkan entitas user yang menggunakan sistem. User dibuat dengan struct custom yang mengimplementasikan ADT SetMap, Stack, dan LinkedList. User terdiri atas 6 komponen, yakni name, password, money, keranjang, riwayat_pembelian, dan wishlist. Komponen name dan password dibentuk dengan tipe data array of character statis untuk menyimpan identitas user. Komponen money dibentuk dengan tipe data primitif integer untuk menyimpan uang yang dimiliki user.

Komponen keranjang dibentuk dengan ADT SetMap yang akan digunakan untuk mengumpulkan barang yang direncanakan untuk dibeli user. Komponen riwayat_pembelian dibentuk dengan ADT Stack yang akan digunakan untuk menyimpan informasi pembelian barang yang pernah dilakukan user, yakni nama barang dan total harga yang dibayarkan. Komponen wishlist dibentuk dengan ADT wishlist yang digunakan untuk menyimpan daftar nama barang yang diinginkan user. Pembuatan ADT dengan struct User disebabkan keperluan untuk menyimpan data user dalam bentuk yang berbeda - beda, termasuk useran ADT Stack, SetMap, dan Linked List yang bukan tipe data primitif.

3.2 *Array*

File `array.h` mengimplementasikan ADT Array Statis, di mana array memiliki ukuran tetap yang telah ditentukan sebelumnya, dan jumlah elemen yang dapat disimpan dibatasi. Struktur data dan userannya tidak berubah dari userannya di Tugas Besar 1. Namun, ADT ini penting untuk digunakan dalam penyimpanan daftar user pada sistem dan mampu mengakomodasi perubahan tipe data pada ADT User. Pada ADT ini, array digunakan untuk menyimpan data user yang terdiri dari nama, kata sandi, saldo, riwayat pembelian, dan wishlist dalam bentuk struktur User.

3.3 *Stack*

File `stack.c` dan `stack.h` mengimplementasikan struktur data Stack dengan karakteristik eksplisit dan alokasi statik. Komponen dari struktur data Stack yang kami gunakan adalah array of barang_dibeli, TOP, dan size. Struktur barang_dibeli merupakan struktur data tambahan untuk merepresentasikan informasi nama barang dengan tipe data array of characters dan total harga yang dibayarkan dengan tipe data integer. Struktur barang_dibeli diperlukan untuk merepresentasikan informasi barang dalam daftar riwayat pembelian

user. Atribut TOP dengan tipe data integer merepresentasikan index dari elemen paling atas dalam sebuah Stack dan digunakan untuk mengetahui lokasi elemen paling atas dalam array of barang_dibeli. Atribut size dengan tipe data integer digunakan untuk mencatat jumlah elemen dalam sebuah Stack.

ADT Stack digunakan untuk memodelkan riwayat pembelian pada file history.c untuk perintah HISTORY <n> dan memodifikasi Stack riwayat_pembelian user pada file cart.c untuk perintah CART PAY. Pada perintah tersebut, program dapat memperlihatkan daftar riwayat pembelian user dimulai dari riwayat yang terbaru, yakni informasi barang_dibeli pada elemen paling atas Stack riwayat_pembelian user. Pada file history.c, digunakan fungsi IsEmptyStack untuk memeriksa kosong atau tidaknya Stack riwayat_pembelian user dan atribut TOP untuk melakukan pencarian informasi barang yang ingin ditampilkan. Pada file cart.c, digunakan fungsi PushStack untuk memasukkan informasi barang yang berhasil dibeli oleh user ke riwayat pembeliannya. Pada file start.c, load.c, dan save.c, struktur data Stack juga digunakan untuk membaca file save atau konfigurasi maupun menyimpan kondisi riwayat pembelian user.

Penerapan prinsip Last-In-First-Out (LIFO) membuat pembelian terbaru yang dimasukkan daftar riwayat akan selalu berada pada posisi paling atas. Dengan begitu, userannya memberikan kemudahan untuk mengetahui urutan pembelian yang dilakukan oleh user.

3.4 *Setmap*

File map.c dan map.h mengimplementasikan struktur data Map dengan setiap atribut key selalu unik, memimik ADT Set. Komponen dari struktur data Setmap adalah Elements dan Count. Elements memiliki tipe array of infotype, array ini dialokasikan secara statik dengan kapasitas maksimum 100 infotype. Infotype sendiri adalah struktur data kustom dengan 2 komponen, yakni key dan value. Key merupakan atribut unik yang digunakan untuk mengidentifikasi atribut Value. Oleh karena itu, pada fungsi InsertMap, dilakukan pengecekan terlebih dahulu dengan fungsi IsMemberMap agar infotype yang dimasukkan tidak memiliki key yang sama dengan infotype yang telah dimasukkan sebelumnya. Adapun atribut Count dengan tipe integer yang merepresentasikan jumlah elemen efektif dalam array Elements.

ADT Setmap digunakan untuk memodelkan keranjang user, yakni barang - barang yang ingin dibeli user untuk kemudian dibayarkan. Tipe data infotype

digunakan untuk merepresentasikan tiap jenis barang yang diinginkan melalui atribut Key dan kuantitas barang yang ingin dibeli melalui atribut Value. Pada file `cart.c`, ADT Setmap digunakan untuk menambahkan, mengurangi, menghapus, maupun menampilkan barang dari keranjang belanja user melalui fungsi `cartadd`, `cartpay`, `cartremove`, `cartshow`, dan `cartfunction`.

Untuk perintah `CART ADD <nama barang> <n>`, akan dilakukan pemeriksaan sudah ada atau tidaknya nama barang yang dimasukkan user pada keranjang belanja melalui pengecekan nama barang sebagai atribut Key. Hanya barang dengan nama barang yang tidak ditemukan dalam daftar Key pada keranjang belanja yang akan dimasukkan ke keranjang belanja. Bila tidak ditemukan, maka akan dimasukkan tipe data infotype baru ke dalam keranjang belanja, dengan nama barang sebagai Key dan `n` (jumlah barang) sebagai Value.

Untuk perintah `CART REMOVE <nama barang> <n>`, akan dilakukan pencarian Key yang sama dengan masukkan nama barang, lalu dilakukan pengurangan atribut Value dari infotype dengan Key tersebut sebanyak `n`. Untuk perintah `CART PAY`, akan dimunculkan tampilan seluruh informasi tentang keranjang belanja user dengan tambahan pencatatan total harga barang paling tinggi dari perkalian atribut Value dengan harga barang dan urutan dalam Lexical Order yang lebih tinggi berdasarkan atribut Key dari barang.

Pengorganisasian data berdasarkan atribut Key yang unik dan kebutuhan untuk mengasosiasikan atribut Value dengan setiap atribut Key memberikan kemudahan untuk modifikasi keranjang belanja user. Pencarian nama barang, pemunculan kuantitas barang, dan pemeriksaan Key unik mudah dilakukan karena akses ke tiap elemen Map dapat dilakukan berdasarkan nama barang. Oleh karena itu, ADT Setmap cocok dengan mekanisme perintah `CART ADD` dan `CART REMOVE` yang memakai masukkan nama barang dan kuantitas barang.

3.5 Linked List

File `listlinier.c` dan `listlinier.h` mengimplementasikan struktur data Linked List. Komponen dari struktur data Linked List yang kami gunakan adalah atribut `First` yang merepresentasikan alamat dari elemen pertama dalam sebuah Linked List bernama `List`. Untuk setiap alamat, akan merujuk pada sebuah elemen `tElmtList`. Setiap `tElmtList` didefinisikan dalam sebuah struct kustom yang dinamakan `ElmtList` dengan 2 komponen, yakni `info` dengan tipe data `nama_barang` dan `next` dengan tipe data `address`. Atribut `nama_barang` merupakan array of characters yang digunakan untuk merepresentasikan nama

barang, sedangkan atribut next merepresentasikan alamat dari elemen selanjutnya dalam suatu Linked List.

ADT Linked List digunakan untuk memodelkan wishlist user, yakni daftar barang yang diinginkan user tanpa pembelian segera. ADT ini digunakan untuk perintah WISHLIST ADD, WISHLIST SWAP <i><j>, WISHLIST REMOVE <i>, WISHLIST REMOVE, WISHLIST CLEAR, dan WISHLIST SHOW yang memodifikasi atau menampilkan wishlist user.

Pengorganisasian data berdasarkan address dari tiap elemen memberikan kemudahan untuk melakukan penambahan, penghapusan, dan penukaran elemen pada wishlist. Alokasi dinamis elemen dari Linked List memberikan fleksibilitas untuk menambahkan elemen baru. Dengan begitu, alokasi elemen baru tidak terpaku pada jumlah elemen maksimum yang dapat disimpan seperti alokasi statik. Selain itu, penghapusan dan pertukaran elemen tidak memerlukan pergeseran elemen lainnya karena hanya perlu memodifikasi atribut First pada List dan atribut next dari elemen ElmtList. Hal ini memberikan efisiensi terhadap kode yang digunakan karena tidak perlu melakukan algoritma pergeseran elemen seperti alokasi dan dealokasi pada struktur data kontigu.

4 Program Utama

Ketika PURRMART pertama kali dijalankan, sistem menampilkan *landing page* berisi ASCII art yang diambil dari banner.txt dan meminta user memasukkan perintah awal START, LOAD, atau QUIT. Command START digunakan untuk memulai permainan dengan memuat file config default yang telah ditentukan di dalam sistem. Begitu perintah ini dieksekusi, PURRMART akan membaca data barang yang tersedia di toko, informasi user yang sudah terdaftar, beserta riwayat dan wishlist yang dimiliki setiap user. Setelah proses pembacaan selesai, user diizinkan untuk memasukkan perintah berikutnya, yakni REGISTER atau LOGIN. Dengan demikian, command START berperan sebagai pintu gerbang utama agar sistem siap digunakan berdasarkan konfigurasi bawaan.

Bila user ingin memuat data dari *file config* tertentu, maka perintah LOAD <filename.txt> dijalankan. Sistem akan membuka berkas konfigurasi sesuai nama yang diberikan, kemudian membaca seluruh data barang di toko, user, riwayat, serta *wishlist* yang ada. *File config* ini dapat berbeda dari *file config* default, sehingga user leluasa menyimpan dan memuat berkas khusus sesuai

kebutuhan. Setelah data berhasil diambil, user dapat langsung menggunakan command lain layaknya setelah mengeksekusi **START**.

Perintah **SAVE <filename.txt>** bertugas untuk menyimpan seluruh perubahan yang terjadi selama sesi permainan ke dalam sebuah berkas konfigurasi. Saat command ini dijalankan, PURRMART akan menuliskan data barang toko, daftar user, riwayat pembelian, dan isi wishlist masing-masing user ke filename.txt yang ditentukan. Dengan command ini, user dapat memastikan bahwa semua penambahan, penghapusan, maupun modifikasi data tidak hilang ketika sistem dimatikan. Apabila command **SAVE** dipanggil tanpa menyebutkan nama berkas, sistem akan menggunakan file config default sebagaimana yang digunakan oleh **START**.

Saat user sudah selesai menggunakan PURRMART, mereka dapat mengetikkan perintah **QUIT** untuk menghentikan program. Namun, sebelum benar-benar keluar, sistem akan menawarkan opsi untuk menyimpan data terbaru. Jika user memilih “Y” atau “y” (Yes), maka program akan menjalankan proses **SAVE** pada file config default. Bila user menolak atau menekan “N” (No), maka program segera menutup tanpa menyimpan perubahan apa pun. Dengan demikian, command ini menjadi penutup keseluruhan alur useran PURRMART.

Command **PROFILE** hanya dapat dipanggil jika user sudah login. Begitu eksekusi dimulai, PURRMART akan menampilkan data pribadi user seperti nama dan saldo saat ini. Informasi tersebut diambil dari data user aktif yang dimuat ketika perintah **LOGIN** dipanggil atau ketika data user dibaca melalui **START/LOAD**. Setelah profil ditampilkan, sistem kembali ke menu utama untuk menunggu instruksi berikutnya.

user yang sudah login dapat mengelola keranjang belanja menggunakan perintah **CART ADD <nama> <n>**. Sistem akan memeriksa apakah barang <nama> tersedia di toko. Jika tersedia, sistem menambahkan <n> unit barang tersebut ke keranjang belanja. Apabila barang tidak ditemukan di toko, maka sistem menampilkan pesan kesalahan. Command ini mempermudah proses belanja karena user dapat menumpuk beberapa barang sebelum melakukan pembayaran.

Apabila user memutuskan untuk mengurangi atau menghapus sebagian barang tertentu dari keranjang, maka perintah **CART REMOVE <nama> <n>** dipakai. Sistem pertama-tama memeriksa apakah barang <nama> ada di keranjang dan apakah kuantitas barang di keranjang mencukupi untuk dikurangi sejumlah <n>. Jika validasi lolos, sistem akan mengurangi kuantitas barang tersebut di

keranjang. Jika kuantitas tidak mencukupi atau barangnya tidak ada, maka perintah gagal dijalankan dan user kembali ke menu utama dengan pesan kesalahan.

Dengan memanggil CART SHOW, user dapat melihat isi keranjang belanja saat ini, terdiri atas nama barang, kuantitas, serta total harga per barang. PURRMART juga menghitung dan menampilkan jumlah biaya keseluruhan yang perlu dikeluarkan apabila user memutuskan untuk membayar semua isi keranjang. Jika keranjang kosong, sistem akan menegaskan bahwa belum ada barang apa pun di dalamnya. Setelah menampilkan informasi, sistem kembali menunggu command selanjutnya.

Setelah yakin dengan isi keranjang, user dapat melakukan pembayaran dengan CART PAY. Sistem akan menampilkan isi keranjang beserta total biaya keseluruhan. user lalu ditanya apakah ingin melanjutkan pembelian. Jika menjawab “Ya” dan saldo mencukupi, sistem akan mengurangi saldo user dan menyimpan riwayat pembelian. Pada riwayat ini, hanya barang dengan total harga paling besar (atau pemenang leksikal jika ada nilai sama) yang dicatat bersama biaya pembeliannya. Jika user tidak memiliki cukup uang atau memilih “Tidak”, pembelian dibatalkan. Proses pun berakhir dan PURRMART kembali ke menu utama.

Command HISTORY <n> berfungsi menampilkan catatan pembelian terbaru hingga sebanyak n kali. Riwayat ditampilkan dengan urutan paling baru di bagian paling atas. Jika jumlah total riwayat kurang dari n, maka semua catatan pembelian akan ditampilkan. Sistem akan menuliskan nama barang serta total biaya yang dikeluarkan. Jika ternyata user belum pernah membeli barang apa pun, sistem menyatakan bahwa riwayat pembelian masih kosong.

Bagi user yang belum ingin langsung membeli barang, PURRMART menyediakan command WISHLIST ADD agar barang tertentu dapat dicatat di daftar keinginan. Sistem akan meminta user memasukkan nama barang yang ingin ditambahkan. Jika barang itu ada di toko dan belum ada di wishlist, barang akan ditambahkan. Apabila sudah ada, PURRMART menginformasikan bahwa barang telah tersedia di wishlist. Jika nama barang yang dimasukkan tidak ditemukan di toko, user mendapat pesan kegagalan.

Untuk mengatur prioritas atau sekadar menukar urutan barang di wishlist, user bisa menggunakan WISHLIST SWAP <i> <j>. Di sini, sistem memeriksa apakah posisi <i> dan <j> valid (ada di wishlist). Jika valid, barang pada posisi <i> ditukar dengan barang pada posisi <j>. Namun, jika salah satu posisi tidak ada

di wishlist, sistem menampilkan pesan kegagalan dan perintah tidak dijalankan. Setelahnya, PURRMART kembali menunggu instruksi selanjutnya.

Perintah `WISHLIST REMOVE <i>` dipakai jika user ingin menghapus barang pada posisi ke-*i*. Pertama, sistem memeriksa apakah wishlist memiliki indeks *i*. Jika ada, barang di posisi tersebut dihapus, dan PURRMART menampilkan pesan bahwa penghapusan berhasil. Jika wishlist kosong atau jumlah barang di dalamnya kurang dari *i*, maka barang gagal dihapus, dan sistem kembali dengan pesan kesalahan.

Selain menghapus berdasar indeks, user dapat menghapus barang wishlist dengan menyebutkan namanya lewat command `WISHLIST REMOVE`. PURRMART akan meminta nama barang yang ingin dihapus. Jika barang tersebut ada dalam wishlist, sistem akan menghapus barang tersebut dan menampilkan notifikasi bahwa proses berhasil. Jika tidak ditemukan, perintah gagal, dan user diberi tahu bahwa penghapusan tidak dapat dilakukan. Command ini memberi keleluasaan bagi user yang lebih mengingat nama barang ketimbang posisinya.

Saat user menginginkan wishlist benar-benar kosong, command `WISHLIST CLEAR` adalah solusinya. Sistem langsung menghapus semua barang yang ada dalam daftar keinginan tanpa konfirmasi lebih lanjut. Setelah pembersihan selesai, PURRMART mengonfirmasi bahwa wishlist telah dikosongkan. Dengan fitur ini, user bisa mengatur ulang wishlist mereka dari awal.

Jika user ingin memeriksa daftar barang apa saja yang sudah masuk wishlist, mereka cukup mengetikkan command `WISHLIST SHOW`. Sistem akan menampilkan barang-barang dalam urutan sesuai penambahan atau posisi saat ini, disertai nomor urut setiap barang. Apabila wishlist kosong, maka program akan memberikan pemberitahuan bahwa belum ada barang apa pun yang dicatat. Setelah itu, control flow kembali ke menu utama untuk menunggu command selanjutnya.

Untuk menampilkan barang-barang yang ada di toko beserta harganya, PURRMART menyediakan perintah `STORE LIST`. Sistem mencetak nama setiap barang yang tersedia dan harganya dalam format `<Nama Barang> - Harga: <Harga>`. Jika toko ternyata masih kosong, maka akan tampil pesan bahwa tidak ada barang apa pun yang sedang dijual. Command ini membantu user mengetahui pilihan barang sebelum memutuskan untuk menambahkannya ke keranjang belanja atau wishlist.

5 Algoritma-Algoritma Menarik

Pada pengerjaan setiap spesifikasi yang ada pada PURRMART, kami menemukan beberapa algoritma menarik. Kami mendefinisikan algoritma menarik sebagai algoritma baru dan unik yang tidak terdapat di dalam spesifikasi. Algoritma ini kemudian menantang kami untuk mengerjakan dengan semangat sebab konsepnya yang menantang.

5.1 Needleman-Wunsch

```
int needleman_wunsch[51][51];
for (int j = 0; j<=panjang_referensi; j++){
    needleman_wunsch[0][j] = 0 + j*GAP;
}
for (int j = 0; j<=panjang_query; j++){
    needleman_wunsch[j][0] = 0 + j*GAP;
}
for(int j = 1; j<=panjang_referensi; j++){
    for(int k = 1; k<=panjang_query; k++){
        needleman_wunsch[k][j] = maksi(needleman_wunsch[k-1][j] + GAP,
                                         needleman_wunsch[k][j-1] + GAP,
                                         needleman_wunsch[k-1][j-1] +
                                         matchatautidak(referensi[j-1],
                                         query[k-1]));
    }
}
```

Algoritma Needleman_Wunsch adalah algoritma *global alignment* yang digunakan untuk *aligning* dua rantai atau sekuens agar kedua rantai atau sekuens tersebut bisa selaras secara holistik dengan cara yang paling optimal. Dalam konteks ini, optimal berarti memiliki skor yang paling besar. Skor ditentukan oleh berapa banyak MATCH, GAP, dan MISMATCH yang dimiliki, serta nilainya dapat secara bebas dipilih.

Algoritma ini memanfaatkan pendekatan pemrograman yang cukup menantang, yaitu *dynamic programming*. Algoritma dimulai dengan membuat suatu matriks 2D untuk menyimpan skor dari *alignment*, lalu baris dan kolom pertama diisi dengan penalti untuk GAP. Lalu, tiap elemen dari matriks ini diisi dengan nilai maksimal dari ketiga kolom yang ada di atas, kiri, dan diagonal kiri atas yang telah dikalkulasi dengan GAP/MATCH/MISMATCH yang dimiliki. Setelah matriks diisi, dilakukan *tracing* dari elemen paling terakhir ke awal agar dapat ditemukan *alignment* terbaik. Dengan ini, bisa didapatkan skor paling efisien dalam *aligning* dua sekuens dan dalam konteks Tugas Besar ini, dapat

digunakan untuk mencari kesamaan dua sekuens senjata biologis di fungsi GLOBALALIGNMENT.

5.2 Backtracking

```
void cari_rute(int jarak, Din_matrix graf, int node_sekarang, int
jumlah_node_didatangi, int node){
    if(jumlah_node_didatangi == node){
        if(jarak < jarak_terkecil || jarak_terkecil == -1){
            jarak_terkecil = jarak;
            for(int i = 0; i < node; i++){
                rute_terbaik[i] = rute[i];
            }
        }
        return;
    }
    for(int i = 0; i < node; i++){
        if(kunjungi[i] == 0 && graf[node_sekarang][i] != -1 && i != node_sekarang){
            kunjungi[i] = 1;
            rute[jumlah_node_didatangi] = i;
            cari_rute(jarak + graf[node_sekarang][i], graf, i,
                jumlah_node_didatangi + 1, node);
            kunjungi[i] = 0;
        }
    }
    return;
}
```

Algoritma Backtracking adalah algoritma dalam penyelesaian suatu masalah dengan cara mencoba semua kemungkinan solusi secara sistematis. Algoritma ini disebut *backtracking* karena algoritma akan mundur atau *backtrack* apabila solusi tidak valid dan dicoba jalur lain. Secara simpel, algoritma ini memilih satu langkah sebagai langkah pertama untuk memulai dan mengikuti langkah itu hingga mencapai suatu solusi.

Apabila solusi yang diberi tidak valid, algoritma akan mundur ke *step* sebelumnya dan coba kemungkinan lain. Apabila solusi yang diberi valid, solusi akan disimpan dan akan dicari juga solusi lain yang lebih efisien atau efektif. Dalam konteks Tugas Besar ini, algoritma ini digunakan pada fungsi OPTIMASIRUTE, yaitu dalam pencarian rute.

Pertama, fungsi `cari_rute` akan dipanggil terus secara rekursif untuk mencoba semua kemungkinan rute dari tiap-tiap *node*. Ketika *node* sudah tidak memiliki tetangga dan masih belum semua *node* dikunjungi, algoritma akan melakukan *backtrack* dan mencari kemungkinan rute lain. Sedangkan ketika semua *node* telah dikunjungi dan kebetulan jaraknya lebih kecil dari jarak terkecil dari

rute-rute sebelumnya, rute yang sekarang akan disimpan sebagai solusi yang paling baik. Dengan algoritma ini, semua rute dicoba secara efisien sehingga dapat dipastikan bahwa rute yang dihasilkan adalah rute yang memiliki jarak total paling kecil.

5.3 keepLastWord dan removeLastWord

Algoritma fungsi keepLastWord dan removeLastWord digunakan pada pemrosesan fungsi CART ADD. Sebelumnya, program yang kami buat menggunakan sebuah ADT Mesin Frasa untuk menerima setiap masukan dari pengguna (membaca satu frasa penuh), sehingga untuk memproses fungsi CART ADD yang memiliki format “CART ADD <nama barang> <jumlah barang>”, dibutuhkan algoritma khusus untuk memotong sebuah frasa menjadi beberapa kata sesuai kebutuhan.

Ketika pengguna memberi masukan “CART ADD <nama barang> <jumlah barang>”, program akan mengecek masukan “CART” dengan fungsi check_str terlebih dahulu, kemudian frasa akan dipotong menjadi “ADD <nama barang> <jumlah barang>” dengan fungsi removeFirstnString. Lalu dilakukan hal yang sama untuk pengecekan masukan “ADD”, sehingga frasa hanya tersisa “<nama barang> <jumlah barang>” saja.

Fungsi removeLastWord berfungsi untuk memperoleh string <nama barang>, sedangkan fungsi keepLastWord berfungsi untuk memperoleh string <jumlah barang>. Fungsi removeLastWord bekerja dengan cara iterasi dari bagian belakang string, lalu mencari spasi terakhir pada string. Jika ditemukan, maka string akan dipotong hingga posisi spasi tersebut kemudian diakhiri dengan null terminator (\0). Sedangkan fungsi keepLastWord bekerja dengan iterasi mulai dari awal string, mencari spasi terakhir pada string, lalu menggeser semua string tersisa yang terletak setelah spasi terakhir ke bagian awal string. Sehingga, selain mengatasi kondisi masukan nama barang yang hanya satu kata saja, program kami dapat mengatasi kondisi dimana masukan nama barang yang memiliki lebih dari satu kata.

6 Data Test

Pada bagian ini, akan disajikan beberapa data uji yang digunakan untuk menguji berbagai fitur dari program yang telah dikembangkan. Setiap data uji dirancang untuk menguji fitur tertentu dalam program dan diharapkan menghasilkan keluaran yang sesuai dengan yang diinginkan. Penjelasan untuk tiap data uji akan mencakup fitur apa saja yang diuji, hasil yang seharusnya

diberikan oleh sistem, serta informasi tambahan yang relevan jika diperlukan. Data uji yang diberikan akan disertai dengan input yang digunakan, proses yang dilakukan oleh sistem, dan output yang dihasilkan sebagai hasil dari eksekusi program. Dengan data uji ini, dapat dievaluasi apakah sistem berfungsi sesuai dengan ekspektasi dan memenuhi persyaratan yang ditetapkan.

6.1 Data Test *Profile*

Pengujian ini berfungsi untuk menguji fitur **Profile** dengan mengambil data dari user.txt. Setelah memasukkan perintah "START", user dapat melihat data diri yang dimiliki oleh seorang user, misalnya data Nama dan Saldo. *Command Profile* baru dapat dijalankan jika user telah melakukan login terlebih dahulu

```
Masukkan command: PROFILE
Nama      : user1
Saldo     : 100
```

Gambar 1. Tampilan Profile ketika dijalankan

6.2 Data Test *Cart Add <nama> <n>*

Command Cart Add pada dasarnya digunakan untuk menambahkan barang ke dalam keranjang *user* dengan jumlah tertentu. Dengan menjalankan *command* ini, user kemudian juga akan diminta inputan nama barang serta jumlah yang ingin ditambahkan. Jika barang valid dan terdapat di dalam toko, maka kemudian barang akan berhasil ditambahkan. Begitu sebaliknya, jika barang tidak terdapat di dalam toko, maka barang akan gagal dimasukkan ke dalam keranjang belanja.

```
Berhasil menambahkan 1 AK47 ke keranjang belanja!
Masukkan command: CART ADD Ayam Goreng Crisbar 1000
Berhasil menambahkan 1000 Ayam Goreng Crisbar ke keranjang belanja!
```

Gambar 2. Tampilan Cart Add ketika berhasil dijalankan

```
Masukkan command: CART ADD Barang apa ini 1
Barang tidak ada di toko!
```

Gambar 3. Tampilan Cart Add ketika barang tidak ada di toko

```
Masukkan command: CART ADD AK47 -10
Barang yang dimasukkan ke dalam keranjang harus berjumlah positif!
Masukkan command: CART ADD AK47 0
```

Gambar 4. Tampilan Cart Add ketika inputan tidak sesuai ketentuan

6.3 Data Test *Cart Remove* <nama> <n>

CART REMOVE digunakan untuk mengurangi jumlah suatu barang tertentu yang sebelumnya telah ditambahkan ke dalam keranjang belanja user. Dengan menjalankan *command* ini, user akan diminta inputan nama barang serta jumlah yang ingin dikurangi. Jika barang tersebut terdapat dalam keranjang dan jumlah yang ingin dikurangi masih dalam batas jumlah yang dimiliki di keranjang, maka perintah ini akan berhasil dijalankan, dan kuantitas barang tersebut di dalam keranjang akan berkurang. Sebaliknya, jika barang tidak ditemukan di dalam keranjang atau jumlah yang ingin dikurangi melebihi kuantitas yang tersedia, maka perintah akan gagal dan user akan mendapatkan notifikasi bahwa proses pengurangan barang tidak dapat dilakukan. Setelahnya, user akan kembali ke menu utama untuk melanjutkan aktivitas selanjutnya.

```
Masukkan command: CART REMOVE AK47 1
Berhasil mengurangi 1 AK47 dari keranjang belanja!
Masukkan command: CART REMOVE Ayam Goreng Crisbar 1
Berhasil mengurangi 1 Ayam Goreng Crisbar dari keranjang belanja!
```

Gambar 5. Tampilan Cart Remove <nama> <n> ketika berhasil dijalankan

```
Masukkan command: CART REMOVE AK47 10
Tidak berhasil mengurangi, hanya terdapat 1 AK47 pada keranjang!
```

Gambar 6. Tampilan Cart Remove <nama> <n> ketika jumlah barang tidak memenuhi

```
Masukkan command: CART REMOVE AK47 -10
Tidak berhasil mengurangi, masukan angka harus positif!
Masukkan command: CART REMOVE AK47 0
Tidak berhasil mengurangi, masukan angka harus positif!
```

Gambar 7. Tampilan Cart Remove <nama> <n> ketika inputan tidak sesuai ketentuan

6.4 Data Test *Cart Show*

CART SHOW digunakan untuk menampilkan seluruh barang yang telah ditambahkan ke dalam keranjang beserta kuantitas masing-masing dan total

biaya yang harus dikeluarkan. Dengan menjalankan *command* ini, sistem akan melakukan pengecekan terhadap isi keranjang belanja user. Jika terdapat barang-barang di dalam keranjang, maka informasi tersebut akan disajikan. Di akhir daftar, akan ditampilkan pula jumlah biaya keseluruhan yang harus dikeluarkan. Namun, jika keranjang dalam keadaan kosong, sistem akan memberikan notifikasi bahwa keranjang tersebut tidak memiliki isi. Setelah pengoperasian *command* ini, user akan kembali ke menu utama untuk melanjutkan aktivitas.

```
Masukkan command: CART SHOW
Berikut adalah isi keranjang belanjamu!
Kuantitas      | Nama                | Total
1009            | Ayam Goreng Crisbar | 20180
13              | AK47                 | 130
1               | Meong                | 500
3               | Lalabu               | 60
Total biaya yang harus dikeluarkan adalah 20870.
```

Gambar 8. Tampilan Cart Show ketika dijalankan dan menampilkan semua cart

```
Masukkan command: CART SHOW
Keranjang kamu kosong!
```

Gambar 9. Tampilan Cart Show ketika isi cart kosong

6.5 Data Test *Cart Pay*

CART PAY digunakan untuk membeli seluruh isi keranjang, dengan syarat uang user mencukupi. Command ini menampilkan barang, kuantitas, serta total harga yang harus dibayar. user diminta konfirmasi (Ya/Tidak):

- Jika “Ya” dan uang cukup, transaksi berhasil, uang berkurang, riwayat pembelian bertambah (menyimpan barang dengan total harga tertinggi, atau yang leksikal terbesar jika sama).
- Jika uang kurang, respon “Tidak”, respon selain “Ya/Tidak”, atau keranjang kosong, transaksi batal.

```
Masukkan command: CART PAY
Kamu akan membeli barang-barang berikut:
Kuantitas | Nama | Total
1 | AK47 | 10
2 | Lalabu | 40
1 | Ayam Goreng Crisbar | 20
Total biaya yang harus dikeluarkan adalah 70, apakah jadi dibeli? (Ya/Tidak):
```

Gambar 10. Tampilan Cart Pay ketika menampilkan semua barang

```
Masukkan command: CART PAY
Kamu akan membeli barang-barang berikut:
Kuantitas | Nama | Total
1 | AK47 | 10
2 | Lalabu | 40
1 | Ayam Goreng Crisbar | 20
Total biaya yang harus dikeluarkan adalah 70, apakah jadi dibeli? (Ya/Tidak): Ya
Selamat kamu telah membeli barang-barang tersebut!
```

Gambar 11. Tampilan Cart Pay ketika ingin membeli barang

```
Masukkan command: CART PAY
Kamu akan membeli barang-barang berikut:
Kuantitas | Nama | Total
1 | AK47 | 10
2 | Lalabu | 40
1 | Ayam Goreng Crisbar | 20
Total biaya yang harus dikeluarkan adalah 70, apakah jadi dibeli? (Ya/Tidak): Tidak
Proses pembelian dibatalkan.
```

Gambar 12. Tampilan Cart Pay ketika barang dibatalkan pembelian

```
Masukkan command: CART PAY
Kamu akan membeli barang-barang berikut:
Kuantitas | Nama | Total
1 | AK47 | 10
2 | Lalabu | 40
1 | Ayam Goreng Crisbar | 20
Total biaya yang harus dikeluarkan adalah 70, apakah jadi dibeli? (Ya/Tidak): Gajelas
Input yang anda masukkan tidak valid!
```

Gambar 13. Tampilan Cart Pay ketika input tidak valid

6.6 Data Test *History* <n>

Command History merupakan *command* yang digunakan untuk menunjukkan riwayat pembelian seorang user. *History* diakses berdasarkan *stack* riwayat

pembelian yang terdapat di setiap user. Riwayat yang dibuat didapat dari setiap barang yang telah dibeli oleh user. Namun, berdasarkan fitur *Cart Pay*, data yang ditampilkan adalah data barang dengan total harga tertinggi di dalam keranjang.

```
Masukkan command: HISTORY 3
Riwayat pembelian barang:
1. Lalabu 70
2. Ayam Goreng Crisbar 20
3. Meong 500
```

Gambar 14. Tampilan History ketika dijalankan dan sesuai inputan masih memuat

```
Masukkan command: HISTORY 10
Riwayat pembelian barang:
1. Lalabu 70
2. Ayam Goreng Crisbar 20
3. Meong 500
4. AK47 10
5. Lalabu 35
6. AK47 100
7. AK47 40
```

Gambar 15. Tampilan History ketika inputan lebih besar dari jumlah barang di keranjang

```
Masukkan command: HISTORY -3
Masukan angka harus bernilai positif!
```

Gambar 16. Tampilan History ketika inputan tidak sesuai

6.7 Data Test *Wishlist Add*

WISHLIST ADD digunakan untuk menambahkan suatu barang ke dalam *wishlist* user. Saat *command* dijalankan, user akan diminta untuk menginputkan nama barang yang ingin ditambahkan. Apabila barang tersebut valid dan belum terdapat di dalam *wishlist*, sistem kemudian akan menambahkan barang

tersebut dengan sukses. Sebaliknya, jika barang tersebut telah ada di *wishlist* sebelumnya, maka user akan diberitahu bahwa barang tersebut sudah terdaftar dan tidak perlu ditambahkan lagi. Jika nama barang yang dimasukkan tidak dikenali atau tidak ada dalam daftar barang yang valid, maka proses penambahan ke wishlist akan gagal. Setelah proses ini, user akan kembali ke menu utama untuk melanjutkan aktivitas lainnya.

```
Masukkan command: WISHLIST ADD  
Masukkan nama barang: LALABU  
Berhasil menambahkan Lalabu ke wishlist!
```

Gambar 17. Tampilan Wishlist Add ketika dijalankan

```
Masukkan command: WISHLIST ADD  
Masukkan nama barang: Balls  
Tidak ada barang dengan nama Balls!
```

Gambar 18. Tampilan Wishlist Add ketika tidak ada barang yang sesuai

```
Masukkan command: WISHLIST ADD  
Masukkan nama barang: AK47  
AK47 sudah ada di wishlist!
```

Gambar 19. Tampilan Wishlist Add ketika barang sudah ada di wishlist

6.8 Data Test *Wishlist Swap* <i><j>

Wishlist Swap digunakan untuk menukar posisi dua barang yang terdapat dalam *wishlist* user berdasarkan indeks posisinya. Saat *command* ini dijalankan, user akan menginput dua posisi (*i* dan *j*) yang ingin ditukar. Jika kedua posisi valid dan terdapat barang pada kedua posisi tersebut, maka proses penukaran akan berhasil dan urutan barang di *wishlist* akan diperbarui. Namun, jika hanya ada satu barang atau posisi tidak valid, maka proses penukaran akan gagal.

```
Masukkan command: WISHLIST SWAP 1 3  
Berhasil menukar posisi Ayam Goreng Crisbar dengan Lalabu pada wishlist!
```

Gambar 20. Tampilan Wishlist Swap ketika dijalankan

```
Masukkan command: WISHLIST SWAP 1 10  
Gagal menukar posisi AK47!
```

Gambar 21. Tampilan Wishlist Swap ketika barang tidak dapat ditukar

```
Masukkan command: WISHLIST SWAP -1 23
Tidak ada barang pada posisi -1 dan/atau 23
```

Gambar 22. Tampilan Wishlist Swap ketika barang tidak ditemukan

6.9 Data Test *Wishlist Remove* <i>

Command ini digunakan untuk menghapus barang pada posisi ke-*i* dari *wishlist*. Saat *command* ini dijalankan, user harus memasukkan posisi barang yang akan dihapus. Jika valid maka barang akan dihapus, sebaliknya maka proses penghapusan akan gagal.

```
Masukkan command: WISHLIST REMOVE 2
Berhasil menghapus barang posisi ke-2 dari wishlist!
```

Gambar 23. Tampilan *Wishlist Remove* <i> ketika dijalankan

```
Masukkan command: WISHLIST REMOVE 69
Penghapusan barang WISHLIST gagal dilakukan, Barang ke-69 tidak ada di WISHLIST!
```

Gambar 24. Tampilan *Wishlist Remove* <i> ketika barang yang diinginkan tidak ada

6.10 Data Test *Wishlist Remove*

WISHLIST REMOVE digunakan untuk menghapus suatu barang dari *wishlist* berdasarkan nama yang diinputkan user. Jika barang tersebut ditemukan dalam *wishlist*, maka barang tersebut akan berhasil dihapus. Namun, jika nama barang yang dimasukkan tidak terdapat di dalam *wishlist*, proses penghapusan akan gagal. Setelah selesai, user akan kembali ke menu utama.

```
Masukkan nama barang yang akan dihapus: LALABU
Lalabu berhasil dihapus dari WISHLIST!
```

Gambar 25. Tampilan *Wishlist Remove* ketika dijalankan

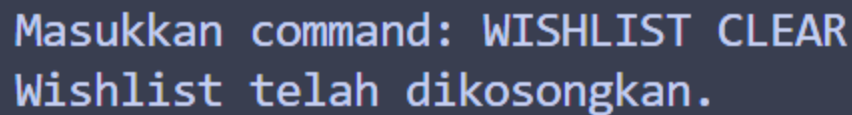
```
Masukkan nama barang yang akan dihapus: Balls
Penghapusan barang WISHLIST gagal dilakukan, Balls tidak ada di WISHLIST!
```

Gambar 26. Tampilan *Wishlist Remove* ketika gagal dijalankan

6.11 Data Test *Wishlist Clear*

Berbeda dengan *wishlist remove*, **WISHLIST CLEAR** dimanfaatkan untuk menghapus semua data barang yang ada di dalam *wishlist* user. Jika user

memasukkan *command Wish Clear*, maka kemudian seluruh *wishlist* akan dikosongkan atau dihapus semuanya.

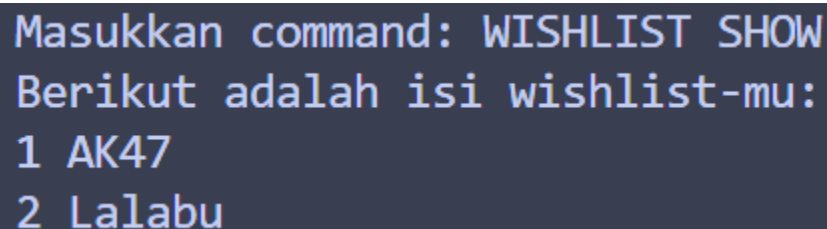


```
Masukkan command: WISHLIST CLEAR
Wishlist telah dikosongkan.
```

Gambar 27. Tampilan *Wishlist Remove* ketika dijalankan

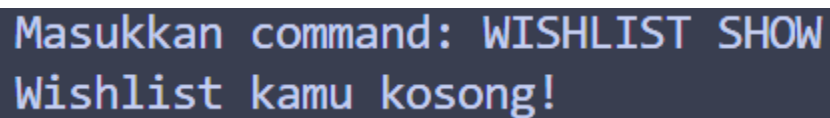
6.12 Data Test *Wishlist Show*

Command *WISHLIST SHOW* merupakan *command* yang bertujuan untuk menampilkan segala *wishlist* yang ada di user. Ketika user memasukkan *command* ini, maka kemudian akan ditampilkan seluruh isi *wishlist* yang dimiliki oleh user.



```
Masukkan command: WISHLIST SHOW
Berikut adalah isi wishlist-mu:
1 AK47
2 Lalabu
```

Gambar 28. Tampilan *Wishlist Show* ketika dijalankan



```
Masukkan command: WISHLIST SHOW
Wishlist kamu kosong!
```

Gambar 29. Tampilan *Wishlist Remove* ketika tidak ada barang *wishlist*

6.13 Data Test Perubahan *Start, Load, dan Save*

Terdapat perubahan file konfigurasi yang menyimpan atribut tambahan pada user, yakni penyimpanan data riwayat pembelian dan *wishlist* masing - masing user. Oleh karena itu, dilakukan perubahan terhadap perintah *START*, *LOAD*, dan *SAVE* untuk mengakomodasi atribut tambahan user tersebut.

Perintah *START* digunakan untuk memuat data file konfigurasi, *config.txt*, sebagai file default dari program. user dapat memasukkan perintah “*START*” ke console dan masuk ke bagian program selanjutnya untuk melakukan login atau register. Jika perintah berhasil dijalankan, akan muncul pesan konfirmasi keberhasilan dan program mampu memuat data nama barang, harga barang, nama user, password user, saldo user, riwayat pembelian user, dan *wishlist* user.

6.14 Data Test Perubahan *Store List*

Terdapat perubahan pada tampilan yang diberikan pada fungsi STORE LIST. Pada awalnya, fungsi STORE LIST hanya menunjukkan *list* barang yang tersedia di toko. Setelah adanya perubahan, fungsi STORE LIST menunjukkan *list* barang beserta harga dari tiap-tiap barang tersebut.

```
Masukkan command: STORE LIST
List barang yang ada di toko :
- AK47 - Harga: 10
- Lalabu - Harga: 20
- Ayam Goreng Crisbar - Harga: 20
- Meong - Harga: 500
```

Gambar 35. Tampilan STORE LIST ketika berhasil dijalankan

6.15 Data Test Bonus *Globalalignment* (Deteksi Kebocoran Senjata Biologis)

Command GLOBALALIGNMENT adalah *command* yang bertujuan untuk membandingkan dua sekuens dengan melakukan *sequence alignment* pada kedua sekuens. Lalu, ditunjukkan apakah terdapat kebocoran pada senjata biologis. Ketika *command* dijalankan, pengguna akan diminta untuk memasukkan sekuens dari referensi. Lalu, user juga akan diminta untuk memasukkan sekuens dari *query*. Setelah memasukkan kedua sekuens, program akan menunjukkan skor hasil *sequence alignment* dan hasil dari penyejajaran sekuens. Lalu, program juga akan menyimpulkan apakah terjadi kebocoran atau tidak pada senjata biologis.

```
Masukkan command: GLOBALALIGNMENT
Masukan sekuens referensi: TAGTAGAATGGGAGAGGTT
Masukan sekuens query: TAGTAGGGTTAATGTT

Skor: 9
Sekuens yang telah disejajarkan:
TAGTAGAATGGGAGAGGTT
TAGTAG---GGTTAATGTT

Yay! Tidak bocor :D
```

Gambar 36. Tampilan GLOBALALIGNMENT ketika tidak ada kebocoran pada sekuens

```

Masukkan command: GLOBALALIGNMENT
Masukan sekuens referensi: TAGTAGAATGGGAGAGGC
Masukan sekuens query:      TAGTAGAATGGGTAAGTC

Skor: 15
Sekuens yang telah disejajarkan:
TAGTAGAATGGGAGAGGC
TAGTAGAATGGGTAAGTC

Oops, ada kebocoran :(

```

Gambar 37. Tampilan GLOBALALIGNMENT ketika ada kebocoran pada sekuens

6.16 Data Test Bonus Optimasirute

Command OPTIMASIRUTE adalah *command* yang bertujuan untuk mencari rute paling efisien dalam pengiriman barang pada berbagai lokasi pengiriman. Ketika *command* dijalankan, user akan diminta untuk memasukkan jumlah lokasi pengiriman atau *node* pada graf. Lalu, user juga akan diminta untuk memasukkan banyak rute atau *edge* antar-*node*. Setelah itu, user pun diminta untuk memasukkan jarak antarlokasi atau *weight* dari *edge* sebanyak jumlah *edge* yang telah dimasukkan sebelumnya. Ketika semua *edge* sudah diberi *weight*, program pun menentukan mana rute yang paling efektif dalam pengiriman barang ke semua lokasi pengiriman. Ketika rute telah ditentukan, program akan mencetak rute tersebut beserta jarak totalnya.

```

Masukkan jumlah lokasi pengiriman (node): 4
Masukkan jumlah rute (edge): 5
Masukkan jarak antarlokasi (weight):
0 1 10
0 2 15
0 3 20
1 2 35
1 3 25

Data diterima, silakan tunggu.

A-ha! Rute paling efektif adalah 0-2-1-3 dengan total jarak 75.

```

Gambar 38. Tampilan OPTIMASIRUTE ketika berhasil dijalankan

7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Profile	Mengetahui keberhasilan command dalam menampilkan profil user	1. Masukkan input berupa PROFILE	PROFILE	user dapat melihat informasi username dan uang/saldo terkait akun yang sedang digunakan	Ditampilkan informasi mengenai user, yang berisi nama dan uang/saldo dari user tersebut
2	Cart Add <nama> <n>	Mengetahui keberhasilan command dalam menambahkan barang ke dalam keranjang	1. Masukkan input berupa CART ADD <NAMA BARANG> <JUMLAH>	CART ADD AK47 1 CART ADD Ayam Goreng Crisbar 1000	user dapat menambahkan barang yang terdapat pada toko ke dalam keranjang	Barang AK47 dan Ayam Goreng Crisbar yang belum ada di keranjang akan masuk ke baris baru di keranjang (dengan jumlah 1 dan 1000). Jika sudah ada, maka jumlah

						barang pada keranjang akan ditambah
3	Cart Remove <nama> <n>	Mengetahui keberhasilan command dalam menghapus barang dari dalam keranjang	1. Masukkan input berupa CART REMOVE <NAMA BARANG> <JUMLAH>	CART REMOVE AK47 1 CART REMOVE Ayam Goreng Crisbar 1	user dapat menghapus atau mengurangi barang dari keranjang	Barang AK47 dihapus dari keranjang dan barang Ayam Goreng Crisbar dikurangi jumlahnya pada keranjang
4	Cart Show	Mengetahui keberhasilan command dalam menampilkan barang dari dalam keranjang	1. Masukkan input berupa CART SHOW	CART SHOW	user dapat melihat isi dari keranjang pada akun yang sedang digunakan	Barang-barang yang ada di keranjang ditampilkan ke layar
5	Cart Pay	Mengetahui keberhasilan command dalam memproses	1. Masukkan input berupa CART PAY 2. Masukkan input kedua berupa "Ya"	CART PAY	user dapat membeli semua barang yang ada pada keranjang	Barang-barang yang ada di keranjang akan "dibeli", maksudnya

		transaksi barang dari dalam keranjang	atau "Tidak"		(jika uang mencukupi) Proses transaksi dibatalkan apabila user memberi input "Tidak"	a dihapus dari keranjang, kemudian barang dengan harga tertinggi pada keranjang akan disimpan ke Stack Riwayat Pembelian.
6	History <n>	Mengetahui keberhasilan command dalam melihat barang-barang yang telah dibeli	1. Masukkan input berupa History <Jumlah yang ingin ditampilkan >	HISTORY 10	user dapat melihat riwayat pembelian barang-barang yang pernah dibeli	Ditampilkan list barang-barang dengan harga tertinggi dari setiap transaksi, dari transaksi terbaru ke transaksi terlama
7	Wishlist Add	Mengetahui keberhasilan command dalam menambahkan barang -	1. Masukkan input WISHLIST ADD 2. Masukkan nama barang yang diinginkan	WISHLIST ADD Masukkan nama barang: Lalabu	user dapat menambahkan barang Lalabu ke wishlist	Barang Lalabu berhasil ditambahkan ke wishlist pada posisi terakhir

		barang yang diinginkan (<i>wishlist</i>)				
8	Wishlist Swap <i> <j>	Mengetahui keberhasilan command dalam menukar barang - barang yang diinginkan (<i>wishlist</i>)	1. Masukkan inputan WISHLIST SWAP <kode barang yang ingin saling ditukar>	WISHLIST SWAP 13	user dapat menukar posisi barang ke-1 dan barang ke-3 di wishlist	Posisi barang ke-1 dan ke-3 berhasil ditukar
9	Wishlist Remove <i>	Mengetahui keberhasilan command dalam menghapus barang ke i dari data yang diinginkan (<i>wishlist</i>)	1. Masukkan inputan WISHLIST REMOVE <index barang yang ingin dihapus>	WISHLIST REMOVE 2	user dapat menghapus barang ke-2 dari wishlist	Barang ke-2 berhasil dihapus dari wishlist user
10	Wishlist Remove	Mengetahui keberhasilan	1. Masukkan inputan WISHLIST REMOVE	WISHLIST REMOVE	user dapat menghapus Lalabu	Lalabu berhasil dihapus

		command dalam menghapus barang dari data yang diinginkan (<i>wishlist</i>)	2. Masukkan inputan nama barang yang ingin di-remove	Masukkan nama barang yang akan dihapus: Lalabu	dari wishlist	dari wishlist
11	Wishlist Clear	Mengetahui keberhasilan command dalam menghapus semua barang dari data yang diinginkan (<i>wishlist</i>)	1. Masukkan inputan berupa WISHLIST CLEAR	WISHLIST CLEAR	user dapat menghapus wishlist	Isi wishlist berhasil dihapus
12	Wishlist Show	Mengetahui keberhasilan command dalam menampilkan semua barang dari	1. Masukkan inputan berupa WISHLIST SHOW	WISHLIST SHOW	user dapat melihat isi wishlist	Semua isi wishlist ditampilkan dengan urutan dan nama barangnya

		data yang diinginkan (<i>wishlist</i>)				
13	Perubahan Start	Mengetahui keberhasilan command dalam membaca file	<ol style="list-style-type: none"> 1. Masukkan perintah untuk ke dalam terminal 2. Masukkan input "START" 	START	File dapat dibaca	<p>Isi file config.txt terbaca ke dalam program</p> <p>(Perubahan terdapat pada pembacaan riwayat pembelian dan wishlist pada file config.txt)</p>
14	Perubahan Load	Mengetahui keberhasilan Load dalam membuka dan mengakses sebuah file	<ol style="list-style-type: none"> 1. Masukkan input berupa LOAD <filename.txt> 	LOAD dummyfile.txt	File dummyfile.txt berhasil dibuka dan diakses jika ada dalam sistem dan memuat data yang tepat	<p>Isi file dummyfile.txt terbaca ke dalam program</p> <p>(Perubahan terdapat pada pembacaan riwayat pembelian dan wishlist pada file dummyfile.txt)</p>

15	Perubahan Save	Mengetahui apakah data terbaru dapat tersimpan di dalam file	1. Panggil SAVE dengan memasukkan inputan SAVE <filename.txt>	SAVE dummyfile.txt	Terdapat data baru untuk Barang, Riwayat Pembelian, dan Wishlist pada dummyfile.txt	Ada data baru untuk Barang, Riwayat Pembelian, dan Wishlist pada dummyfile.txt
16	Perubahan Store List	Mengetahui bahwa barang-barang di dalam toko dapat dilihat	1. Jalankan perintah STORE LIST	STORE LIST	user dapat melihat data barang apa saja yang ada di dalam toko	Program akan menampilkan list nama Barang serta harga Barang pada etalase <i>store</i>
17	Global alignment	Mengetahui keberhasilan command dalam mengecek kesamaan antara sekuens senjata biologis dan sekuens hasil	1. Masukkan Perintah GLOBALALIGNMENT 2. Masukkan sekuens referensi 3. Masukkan sekuens <i>query</i>	GLOBALALIGNMENT TAGTAGA ATGGGAGAGGC TAGTAGA ATGGGTA AGTC	user dapat mengecek kesamaan antara sekuens senjata biologis dengan sekuens hasil metagenomik	Program akan menampilkan skor (dengan ketentuan Match +1, Mismatch 0, Gap penalty -1), dan menampilkan kedua sekuens yang telah disejajarkan, diikuti

		metage nomik				dengan keterangan apakah terjadi kebocoran atau tidak
18	Optim asirute	Menget ahui keberha silan comman d dalam mencari rute yang paling efisien dalam mengiri m suatu barang	<ol style="list-style-type: none"> 1. Masukkan inputan berupa OPTIMASIR UTE 2. Masukkan jumlah lokasi pengiriman /node 3. Masukkan jumlah rute (edge) 4. Masukkan jarak antarlokasi (weight) sesuai banyaknya edge 	OPTIMAS IRUTE 4 5 0 1 10 0 2 15 0 3 20 1 2 35 1 3 25	user dapat menentuk an rute paling efisien dalam pengirima n barang	Program akan menampil kan rangkaian rute yang paling efisien (pada kasus ini 0-2-1-3) beserta total jaraknya (pada kasus ini yaitu 75)

8 Pembagian Kerja dalam Kelompok

Nama Lengkap - NIM	Deskripsi Tugas
Seren Elizabeth Siahaan - 18221160	<ul style="list-style-type: none"> • Mengerjakan kode bagian perubahan Start, Load, Save, dan Quit. • Mengerjakan laporan bagian ringkasan, seluruh Struktur Data (ADT), dan Data Test beserta penjelasan Perubahan Start, Load, dan Save.

Zaka Hanif Nabalah - 18223006	<ul style="list-style-type: none"> • Mengerjakan kode bagian fungsi Wishlist Add, Wishlist Swap <i><j>, Wishlist Remove <i>, Wishlist Remove, Wishlist Clear, Wishlist Show • Mengerjakan laporan bagian Data test dan test Script untuk semua fungsi Wishlist • Melakukan testing program beserta mencari edge case pada program
Rayhan Hidayatul Fikri - 18223022	<ul style="list-style-type: none"> • Mengerjakan kode bagian fungsi Profile, dan History • Mengerjakan sebagian besar bagian laporan • Merapikan keseluruhan laporan
Favian Rafi Laftiyanto - 18223036	<ul style="list-style-type: none"> • Mengerjakan kode bagian fungsi Cart Add <nama> <n>, Cart Remove <nama><n>, Cart Show, Cart Pay , dan mengupdate Help • Mengerjakan laporan bagian Data Test dan Test Script untuk semua fungsi Cart, Profile, History, Globalalignment, Optimasirute dan Perubahan Store List • Mengerjakan laporan bagian Algoritma menarik keepLastWord dan removeLastWord • Melakukan testing program beserta mencari edge case pada program
Wijasara Aptaluhung - 18223088	<ul style="list-style-type: none"> • Mengerjakan kode bagian bonus fungsi Globalalignment, Optimasirute, dan Store List • Mengerjakan keseluruhan ADT • Mengerjakan driver ADT • Mengerjakan laporan bagian Penjelasan Tambahan Spesifikasi Tugas dan Algoritma-Algoritma Menarik

	<ul style="list-style-type: none"> • Mengontak asisten untuk melakukan asistensi dan membuat notulensi
--	---

9 Lampiran

9.1 Deskripsi Tugas Besar

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 user 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.

Command

user dapat memasukkan *command-command* berikut.

a. PROFILE

PROFILE adalah *command* yang digunakan untuk melihat data diri user. PROFILE hanya dapat dipanggil saat status user 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 **user memiliki uang yang cukup** untuk membeli seluruh barang keranjang. Pembelian akan mengurangi uang yang dimiliki user 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. Jika terdapat barang dengan nama Zebra dan AK47 yang memiliki total sama, maka Zebra akan dimasukan ke *history* karena secara lexical $z > a$.

Jika tidak mengerjakan bonus, maka barang yang dimasukan ke history **hanya 1** dengan ketentuan di atas.

f. HISTORY <n>

HISTORY adalah *command* yang digunakan untuk menunjukkan riwayat pembelian seorang user. 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 user.

k. WISHLIST CLEAR

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

I. WISHLIST SHOW

WISHLIST SHOW adalah *command* yang digunakan untuk menunjukkan barang-barang yang sudah dimasukkan ke dalam wishlist.

m. Perubahan Command START, LOAD, dan SAVE

Terdapat perubahan konfigurasi yang harus ditambahkan dalam implementasi *command* START, LOAD, dan SAVE. **BACA KONFIGURASI SISTEM UNTUK PEMBAHARUAN SAVE FILE!**

n. Perubahan Command STORE LIST

STORE LIST akan menampilkan nama barang yang dijual **beserta harganya**.

o. 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 kakas *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





p. 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.

9.2 Notulen Rapat




Asistensi I




Tanggal : Rabu, 20 November 2024	Catatan Asistensi: 1. Kalau memanggil fungsi quit(), itu save config file-nya ke mana, karena kalo pake quit tidak ada keterangan save ke mana, berbeda dengan fungsi save gitu? Bebas mau disimpan di konfig atau mau simpan di file
Tempat : Google Meet	
Kehadiran Anggota Kelompok:  1. Seren Elizabeth Siahaan	

<p>18221160</p>  <p>2. Zaka Hanif Nabalah 18223006</p>  <p>3. Rayhan Hidayatul Fikri 18223022</p>  <p>4. Favian Rafi Laftiyanto 18223036</p>  <p>5. Wijaksana Aptaluhung 18223088</p>	<p>baru, tidak di set dari kami.</p> <p>2. Di wordle, kalo pakai fscanf di work challenge apakah boleh? Tidak apa-apa.</p> <p>3. Berarti fscanf boleh di semua file? Boleh.</p> <p>4. Laporan yang ada di drive itu untuk milestone 2 nanti atau untuk milestone ini aja? Nanti laporan untuk milestone 2 akan berlanjut dari laporan yang sekarang.</p> <p>5. Untuk setiap fungsi apa harus dibuat file headernya? Apakah headernya harus dipisah-pisah apa boleh digabung saja seperti yang di readme di spesifikasi? Ya, harus dibuat header filenya. Boleh digabung, boleh dipisah, yang di spek hanyalah contoh saja.</p> <p>6. Cara nge-run file-nya gimana kalau yang di-include headernya? Misal ingin run runStack.c tapi pakai stack.c nge-include stack.h. Cara runnya:</p>
---	--

	gcc stack.c runStack.c -o test, ./test
	Tanda Tangan Asisten:  Aulia Nadhirah Yasmin B.

Asistensi II

Tanggal : Rabu, 18 Desember 2024	Catatan Asistensi: <ol style="list-style-type: none"> 1. Untuk kerapihan tampilan, apakah masuk nilai? Tidak ya, tidak masuk nilai. 2. Untuk fungsi HELP, kan muncul fungsi-fungsi baru. Nah, dari fungsi HELP M1, apa perlu direvisi dengan ditambah fungsi dari M2? Ya, boleh banget. 3. Di Spesifikasi, ADT User yang keranjang, riwayat pembelian, dan <i>wishlist</i> tidak pakai <i>pointer</i>. Kalau kita tambahkan <i>pointer</i> boleh nggak? Boleh aja. 4. Untuk soal yang graf, dibilang di Spesifikasi kalau tiap <i>node</i> harus terhubung. Namun, di contoh soalnya nggak seperti itu. Itu gimana kak? Itu hanya contoh tampilannya ya. 5. Untuk fungsi Cart kan memunculkan isi Cartnya, nah itu di Spesifikasi dibikin kayak tabel gitu. Itu caranya gimana ya kak? Di tahun kakaknya, ada fungsi
Tempat : Google Meet	
Kehadiran Anggota Kelompok:  1. Seren Elizabeth Siahaan 18221160  2. Zaka Hanif Nabalalah 18223006  3. Rayhan Hidayatul Fikri 18223022	

 4. Favian Rafi Laftiyanto 18223036  5. Wijaksana Aptaluhung 18223088	<p>buat bikin tabel gitu, sedangkan di tahun kami gak ada. Jadi, gak wajib bikin tabel kok, tapi boleh banget kalo dibikin.</p> <p>6. Kalau semua fungsinya dijadiin di satu file, boleh ngga kak? Boleh aja.</p>
	<p>Tanda Tangan Asisten:</p>  Aulia Nadhirah Yasmin B.

9.3 Log Activity Anggota Kelompok

Tanggal	Anggota	Aktivitas
17 Desember 2024	18221160 Seren Elizabeth Siahna	<ul style="list-style-type: none"> Mengganti <i>infotype</i> menjadi nama_barang (list) dan barang_dibeli (stack), nil dan address dibedakan (stack, map), nama fungsi, add startv2.c,

		startv2.h, savev2.c <ul style="list-style-type: none"> • Menambahkan atribut size ke fitur loadv2.c • Melakukan perubahan di fitur startv2.c • Menghapus print di fitur start • <i>Fixing</i> fitur savev2.c • Melakukan perubahan nama fungsi cartremove.c
	18223088 Wijaksana Aptaluhung	<ul style="list-style-type: none"> • Melakukan perubahan nama <i>listlinier.c</i>
	18223036 Favian Rafi Laftiyanto	<ul style="list-style-type: none"> • Membuat versi pertama semua fungsi cart • Memperbaiki pemanggilan nama fungsi • Menambah ADT baru ke MakeFile • Melakukan perubahan nama pada fungsi CopyString • Menambahkan pointer ke ADT Map, Stack, Listi inside User • Menambahkan fungsi Cart dan Wishlist di MakeFile • Melakukan perubahan minor di fungsi Cart Pay

		<ul style="list-style-type: none"> • Melakukan perubahan pada nama fungsi Cart Remove • Melakukan fiksasi pada fungsi load dan loaddriver
	18223006 Zaka Hanif Nabalab	<ul style="list-style-type: none"> • Menambahkan versi pertama dari wishlist • Memperbaiki fitur wishlist • Memperbaiki fitur cart
	18223022 Rayhan Hidayatul Fikri	<ul style="list-style-type: none"> • Menambahkan fitur profile • Perbaiki fitur profile dengan pengaksesan dari stack yang udah dibuat oleh favian
18 Desember 2024	18221160 Seren Elizabeth Siahaan	<ul style="list-style-type: none"> • Menambahkan startv3.c • Memperbaiki command loadv2.c • Memperbaiki command savev2.c • Menambahkan command savedriver.c • Memperbaiki command loaddriver.c • Memperbaiki fungsi save2.c untuk urutan barang di riwayat_pembelian
	18223036 Favian Rafi	<ul style="list-style-type: none"> • Menggabungkan

	Laftiyanto	fungsi Cart ke dalam fungsi main <ul style="list-style-type: none"> • Perubahan minor di <i>bug</i> fungsi Value
19 Desember 2024	18221160 Seren Elizabeth Siahaan	<ul style="list-style-type: none"> • Memperbaiki driver dan dummyfile.txt • Memperbaiki loadv2.c dan mesinkata.c untuk membaca sebuah file logistik • Memperbaiki dengan mencoba membuat atribut baru user • Memperbaiki input untuk command fitur • Menambahkan printf input pada fitur cartremove
	18223088 Wijaksana Aptaluhung	<ul style="list-style-type: none"> • Menambahkan command Globalalignment
	18223036 Favian Rafi Laftiyanto	<ul style="list-style-type: none"> • Menambahkan Globalalignment ke main & help
20 Desember 2024	18221160 Seren Elizabeth Siahaan	<ul style="list-style-type: none"> • Menambahkan pengecekan kondisi pada command save • Memperbaiki input quit dan save • Melakukan perubahan pada read.me
	18223088 Wijaksana Aptaluhung	<ul style="list-style-type: none"> • Menambahkan fungsi bonus Optimasi Rute • Menambahkan

		<p>pengecekan sekuens valid atau tidak</p> <ul style="list-style-type: none"> • Menambahkan driver untuk ADT yang baru
	18223036 Favian Rafi Laftiyanto	<ul style="list-style-type: none"> • Menambahkan Optimasi Rute ke main • Melakukan perubahan minor bug di input command Cart • Menambahkan optimasirute.c ke MakeFile • Menambahkan optimasirute dan wishlist show ke Help • Melakukan perubahan fungsi Cart • Memperbaiki tampilan tabel Cart Show dan Cart Pay • Memperbaiki Cart Add pada kasus masukan angka kosong, angka <i>insensitive</i>
	18223006 Zaka Hanif Nabalab	<ul style="list-style-type: none"> • Melakukan perubahan wishlist • Melakukan perubahan validasi input dari wishlist • Membuat kasus insensitif pada fungsi cart • Menambahkan fungsi strlen ke str

		<ul style="list-style-type: none"> • Melakukan perubahan kondisi quit pada quir
21 Desember 2024	18221160 Seren Elizabeth Siahaan	<ul style="list-style-type: none"> • Melakukan perubahan operator pada quit.c dan savev2.c
	18223088 Wijaksana Aptaluhung	<ul style="list-style-type: none"> • Revisi optimasi rute
22 Desember 2024	18221160 Seren Elizabeth Siahaan	<ul style="list-style-type: none"> • Mengupdate readme • Menambahkan pengecekan kondisi lebih banyak pada load.c • Melakukan perubahan pada dummyfile.txt • Melakukan perubahan pada load.c