

LAPORAN TUGAS BESAR

IF2111 Algoritma dan Struktur Data STI

PURRMART


Dipersiapkan oleh:

Kelompok 2

18221026 Syasya Umaira
18222041 Kezia Karen Cahyadi
18223013 Raka Adhitya Nugraha
18223037 Michelle Hamdani
18223053 Daffa Athalla Rajasa
18223079 Atharizza Muhammad Athaya

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		IF2111-TB-01-II		
		Revisi	1	22 Desember 2024

Daftar Isi

1	Ringkasan.....	3
2	Struktur Data (ADT).....	5
	2.1 Struktur Data Custom.....	5
	2.2 Struktur Data Stack.....	6
	2.3 Struktur Data Map.....	7
	2.4 Struktur Data Linked List.....	8
3	Program Utama.....	9
4	Algoritma-Algoritma Menarik.....	10
	4.1 Algoritma 1.....	10
	4.2 Algoritma 2.....	10
5	Data Test.....	12
	5.1 Data Test START.....	12
	5.2 Data Test LOAD.....	12
	5.3 Data Test SAVE.....	13
	5.4 Data Test STORE LIST.....	13
	5.5 Data Test PROFILE.....	14
	5.6 Data Test CART ADD <nama> <n>.....	14
	5.7 Data Test CART REMOVE <nama> <n>.....	14
	5.8 Data Test CART SHOW.....	15
	5.9 Data Test CART PAY.....	15
	5.10 Data Test HISTORY <n>.....	17
	5.11 Data Test WISHLIST ADD.....	17
	5.12 Data Test WISHLIST SWAP <i> <j>.....	18
	5.13 Data Test WISHLIST REMOVE <i>.....	18
	5.14 Data Test WISHLIST REMOVE.....	19
	5.15 Data Test WISHLIST CLEAR.....	19
	5.16 Data Test WISHLIST SHOW.....	20
6	Test Script.....	20
7	Pembagian Kerja dalam Kelompok.....	24
8	Lampiran.....	24
	8.1 Deskripsi Tugas Besar.....	24
	8.2 Notulen Rapat.....	29
	Asistensi I.....	29
	8.3 Log Activity Anggota Kelompok.....	30

1 Ringkasan

PURRMART adalah sebuah aplikasi e-commerce berbasis command-line interface (CLI) yang dirancang untuk mensimulasikan aktivitas belanja online dengan berbagai fitur manajemen barang. Aplikasi ini memungkinkan pengguna untuk mengelola barang di toko, menyimpan dan membeli barang dari keranjang, membuat dan menghapus wishlist, serta bekerja untuk mendapatkan penghasilan. Dikembangkan menggunakan bahasa pemrograman C, **PURRMART** memanfaatkan library standar seperti `stdio.h`, `stdlib.h`, `time.h`, dan `math.h` untuk memenuhi kebutuhan pengguna dengan efisien.

Aplikasi ini dimulai dengan tiga opsi utama: **START**, **LOAD**, dan **HELP**. Opsi **START** digunakan untuk memuat file konfigurasi default yang berisi informasi akun dan daftar barang di toko. **LOAD** memungkinkan pengguna untuk melanjutkan sesi sebelumnya dengan memuat file save dari folder tertentu, sedangkan **HELP** menampilkan panduan lengkap mengenai *command* yang tersedia.

Setelah memulai aplikasi, pengguna dapat mendaftarkan akun baru dengan atribut berupa username dan password yang terdiri dari satu kata dalam proses **REGISTER**. Lalu, pengguna dapat masuk ke akun di sistem dengan akun yang sudah didaftarkan sebelumnya menggunakan *command* **LOGIN**. Apabila password salah atau username belum didaftarkan, maka pengguna tidak akan berhasil untuk masuk ke dalam sistem. Pengguna dapat melihat data diri saat pengguna telah melakukan login dengan menggunakan *command* **PROFILE**.

Pengguna juga dapat menghasilkan uang dalam aplikasi, dengan memilih pekerjaan tertentu melalui *command* **WORK**, yang membutuhkan durasi tertentu untuk menyelesaikannya. Selama bekerja, aplikasi tidak dapat digunakan. Selain itu, pengguna dapat menyelesaikan tantangan melalui *command* **WORK CHALLENGE** untuk mendapatkan penghasilan tambahan. Pendapatan yang diperoleh dapat digunakan untuk membeli barang di toko atau memenuhi kebutuhan lainnya.

Manajemen barang di toko menjadi salah satu fitur utama. Pengguna dapat melihat daftar barang yang tersedia di toko menggunakan *command* **STORE LIST**. Jika pengguna ingin menambahkan barang baru, mereka dapat mengajukan permintaan dengan *command* **STORE REQUEST**, yang kemudian dapat diproses menjadi barang baru di toko menggunakan *command* **STORE SUPPLY**. Barang yang tidak lagi diperlukan dapat dihapus dengan *command* **STORE REMOVE**.

Selain itu, fitur keranjang belanja memberikan fleksibilitas bagi pengguna untuk menambahkan barang dengan jumlah tertentu ke dalam keranjang melalui *command* **CART ADD**, mengurangi jumlah barang dengan **CART REMOVE**, atau menampilkan isi keranjang dengan **CART SHOW**. Untuk menyelesaikan transaksi, pengguna dapat menggunakan *command* **CART PAY** untuk membeli barang-barang yang ada di keranjang.

PURRMART juga menyediakan fitur riwayat pembelian yang dapat diakses melalui *command* **HISTORY**. Riwayat ini disusun dari transaksi terbaru hingga yang paling lama, memungkinkan

pengguna untuk melihat rekam jejak pembelian mereka. Sementara itu, fitur wishlist memberikan kebebasan bagi pengguna untuk menyimpan barang yang diinginkan, menukar posisi barang dalam daftar, atau menghapus barang berdasarkan posisi atau nama. *Command* seperti **WISHLIST ADD**, **WISHLIST REMOVE**, dan **WISHLIST CLEAR** memberikan fleksibilitas penuh dalam mengelola wishlist.

Pengguna dapat menambahkan suatu barang ke wishlist dengan *command* **WISHLIST ADD**. Lalu, pengguna dapat menukar barang dengan posisi ke-i dengan barang posisi ke-j pada wishlist. Untuk menghapus barang dengan posisi ke-i, dapat digunakan *command* **WISHLIST REMOVE <i>**. Dalam menghapus barang berdasarkan nama barang, digunakan *command* **WISHLIST REMOVE**. Untuk menghapus semua barang, digunakan *command* **WISHLIST CLEAR**. Terakhir, untuk menunjukkan barang yang sudah dimasukkan ke dalam wishlist, digunakan *command* **WISHLIST SHOW**.

Setelah menyelesaikan aktivitas, pengguna dapat keluar dari akun mereka dengan *command* **LOGOUT**. Setelah menggunakan perintah ini, sistem akan memastikan bahwa pengguna telah berhasil keluar dari akun mereka apabila sudah melakukan **LOGIN**. Aplikasi ini juga menyediakan fitur penyimpanan status melalui *command* **SAVE**, yang memungkinkan pengguna untuk melanjutkan sesi mereka di kemudian hari. Untuk keluar dari aplikasi, pengguna dapat menggunakan *command* **QUIT** dengan opsi untuk menyimpan atau tidak menyimpan perubahan sesi terakhir. Terakhir, terkait menu bantuan, terdapat *command* **HELP** untuk menampilkan daftar *command* yang dapat digunakan beserta deskripsi fungsinya, memandu pengguna dalam menggunakan aplikasi.

Pada laporan ini, akan dibahas mengenai spesifikasi dari tugas besar PURRMART yang kami buat, struktur data yang dibuat, persoalan yang diselesaikan, alasan pemilihan dan juga implementasi dari struktur data tersebut. Selain itu, pada laporan ini juga tertera berbagai macam *command* yang terdapat pada program kami beserta *data test* dan juga *test script* untuk semua *command* yang terdapat pada program ini, untuk memastikan setiap fitur berjalan dengan baik. Dengan berbagai fitur yang lengkap dan antarmuka CLI yang sederhana, PURRMART memberikan pengalaman simulasi e-commerce yang efisien, nyaman, dan ramah pengguna.

2 Struktur Data (ADT)

Dalam program ini kami menggunakan beberapa ADT, yaitu ADT User, ADT Barang, ADT Stack, ADT Setmap, dan ADT Linked List.

2.1 Struktur Data Custom

Struktur data *Custom* terdiri dari dua tipe data utama, yakni *User* & *Barang*. *User* merupakan tipe data bentukan yang menyimpan informasi pengguna seperti nama yang terdiri dari maksimal 50 karakter, password yang terdiri dari maksimal 50 karakter, dan uang. Lalu, ada barang

merupakan tipe data bentukan yang menyimpan informasi barang seperti nama yang terdiri dari maksimal 50 karakter dan harga. Primitif yang digunakan untuk program ini adalah:

makeUser(): Fungsi ini digunakan untuk membuat objek *User* baru dengan nama, password, dan saldo uang 0. riwayat pembelian dan wishlist.

makeBarang(): Fungsi ini digunakan untuk membuat objek *Barang* baru dengan nama dan harga barang.

IsMarkUser(): Fungsi ini digunakan untuk mengecek apakah pencarian sudah sampai ke akhir (*Mark User*).

IsSameUser(): Fungsi ini digunakan untuk mengecek apakah user yang dibandingkan sama atau tidak.

IsSameBarang(): Fungsi ini digunakan untuk mengecek apakah barang yang dibandingkan sama atau tidak.

IsSameString(): Fungsi ini digunakan untuk mengecek apakah string yang dibandingkan sama atau tidak.

CopyString(): Fungsi ini digunakan untuk menyalin string yang dibandingkan.

ADT Custom ini digunakan untuk mengelola data pengguna dan data barang dalam sistem PURRMART. Dengan menggunakan ADT ini, informasi pengguna dan barang dapat disimpan dan dikelola dengan cara yang terstruktur dan mudah diakses. Pemilihan ADT Custom dalam PURRMART didasarkan pada kebutuhan untuk menyimpan dan mengelola data pengguna dengan cara yang terstruktur. Diimplementasikan sebagai ADT Custom dengan nama file header “custom.h”.

2.2 Struktur Data Stack

Pada ADT Stack, data disimpan menggunakan prinsip LIFO (Last In, First Out). Artinya, elemen yang terakhir dimasukkan ke dalam stack akan menjadi elemen pertama yang dikeluarkan. Struktur ini cocok digunakan untuk menyimpan data sementara, mencatat riwayat transaksi, atau mengimplementasikan fitur undo-redo pada aplikasi PURRMART. Berikut adalah primitif yang digunakan:

CreateEmptyStack(): Menginisialisasi stack biasa agar kosong dengan mengatur nilai TOP ke Nil (-1).

IsEmptyStack(): Mengecek apakah stack biasa kosong, mengembalikan true jika TOP bernilai Nil.

IsFullStack(): Mengecek apakah stack biasa penuh, mengembalikan true jika TOP mencapai $MaxEl - 1$.

PushStack(): Menambahkan elemen baru ke stack biasa, setelah memastikan stack belum penuh, dengan menempatkan elemen di posisi $TOP + 1$.

PopStack(): Menghapus elemen teratas dari stack biasa, mengembalikan elemen tersebut, dan mengurangi nilai TOP sebesar 1.

FlipStack(): Membalikan urutan elemen teratas dari stack biasa menjadi elemen terbawah, dan elemen terbawah menjadi elemen teratas.

ADT Stack digunakan untuk mengelola data sementara, mencatat riwayat transaksi seperti daftar barang dan total harga, serta mempermudah implementasi fitur undo-redo pada aplikasi. Struktur stack dipilih karena sederhana dan sangat efisien untuk operasi berbasis LIFO. Operasi dasar seperti Push dan Pop memiliki kompleksitas waktu $O(1)$, sehingga pengelolaan data sementara menjadi cepat. Selain itu, struktur ini sesuai untuk skenario di mana elemen terakhir yang dimasukkan harus diakses lebih dulu. Struktur ini diimplementasikan dalam ADT Stack dengan file header bernama 'stack.h'.

2.3 Struktur Data Map

Struktur data Map adalah struktur data yang dirancang untuk menyimpan pasangan key-value dengan properti unik pada setiap key. Struktur ini menggunakan hash table atau tree untuk memastikan bahwa setiap key bersifat unik. Operasi seperti pencarian, penambahan, dan penghapusan dapat dilakukan dengan efisien, dengan kompleksitas waktu mendekati $O(1)$. Map sangat cocok untuk aplikasi yang memerlukan penyimpanan data terorganisir, seperti mencatat daftar barang dalam keranjang dalam aplikasi PURRMART. Berikut adalah primitif yang digunakan:

IsSameStringMap(): Fungsi ini digunakan untuk mengecek apakah string dalam struktur data map yang dibandingkan sama atau tidak.

CreateEmptyMap(): Menginisialisasi map agar kosong dengan mengatur jumlah elemen map ke 0.

IsEmptyMap(): Memeriksa apakah map kosong, mengembalikan true jika jumlah elemen map adalah 0.

IsFullMap(): Memeriksa apakah map penuh, mengembalikan true jika jumlah elemen telah mencapai kapasitas maksimum.

ValueMap(): Mengambil nilai (value) yang terkait dengan key pada map, mengembalikan nilai tersebut jika key ada.

InsertMap(): Menambahkan pasangan key-value ke dalam map, dengan memastikan bahwa map belum penuh dan key belum ada di map.

DeleteMap(): Menghapus pasangan key-value dari map berdasarkan key, dengan memastikan bahwa key tersebut ada di dalam map.

IsMemberMap(): Mengecek apakah sebuah key ada di dalam map, mengembalikan true jika key ditemukan.

DisplayMap(): Mencetak map, termasuk key dan value dari tiap elemen map.

ADT Map digunakan untuk menyelesaikan permasalahan penyimpanan data yang terdiri dari pasangan key-value dengan key yang unik. Struktur ini mempermudah proses pencarian, penambahan, penghapusan, dan pemeriksaan elemen berdasarkan key. ADT ini dipilih karena kemampuannya menyimpan data secara terstruktur dengan memastikan tidak ada duplikasi key. Selain itu, struktur ini mendukung akses data yang efisien, di mana setiap key dapat langsung

digunakan untuk menemukan nilai terkait. Struktur ini diimplementasikan sebagai ADT SetMap dengan file header bernama 'map.h'.

2.4 Struktur Data Linked List

Struktur data *Linked List* merupakan struktur data linear yang terdiri dari serangkaian node yang saling terhubung melalui pointer secara dinamis. Setiap node dalam *linked list* memiliki dua bagian utama, yaitu data yang menyimpan nilai atau informasi, dan pointer/link yang menunjuk ke node berikutnya dalam list. Berbeda dengan array yang memiliki ukuran tetap, linked list tidak memiliki batasan ukuran karena memori dialokasikan secara dinamis sesuai kebutuhan, sehingga lebih fleksibel dalam penggunaan memori. *Linked List* cocok digunakan untuk aplikasi yang memerlukan pengelolaan data secara dinamis, di mana ukuran data tidak diketahui atau dapat berubah-ubah, seperti mencatat daftar *wishlist* dalam aplikasi PURRMART. Berikut adalah primitif yang digunakan:

CopyStringLL(): Fungsi ini digunakan untuk menyalin string yang dibandingkan dalam ADT Linked List.

IsEmptyLinkedList(): Memeriksa apakah *linked list* kosong, mengembalikan true jika jumlah elemen map adalah 0.

CreateEmptyLinkedList(): Menginisialisasi *linked list* agar kosong dengan mengatur jumlah elemen ke 0.

AlokasiLinkedList(): Mengalokasikan memori untuk sebuah elemen baru di *linked list*. Mengembalikan *address* elemen tersebut jika alokasi berhasil, atau nil jika gagal.

DealokasiLinkedList(): Mengembalikan *address* elemen *linked list* ke sistem memori.

SearchLinkedList(): Mencari elemen dengan nilai tertentu dalam *linked list*, mengembalikan *address* elemen tersebut jika ditemukan, atau nil jika tidak ada.

InsVFirstLinkedList(): Menambahkan elemen baru dengan nilai tertentu ke awal *linked list*.

InsVLastLinkedList(): Menambahkan elemen baru dengan nilai tertentu ke akhir *linked list*.

DelVFirstLinkedList(): Menghapus elemen pertama *linked list* dan mengembalikan nilainya.

DelVLastLinkedList(): Menghapus elemen terakhir *linked list* dan mengembalikan nilainya.

InsertFirstLinkedList(): Menambahkan elemen baru (dengan *address* tertentu) ke awal *linked list*.

InsertAfterLinkedList(): Menyisipkan elemen baru setelah elemen tertentu dalam *linked list*.

InsertLastLinkedList(): Menambahkan elemen baru (dengan *address* tertentu) ke akhir *linked list*.

DelFirstLinkedList(): Menghapus elemen pertama *linked list* dan mengembalikan *address*-nya.

DelPLinkedList(): Menghapus elemen *linked list* berdasarkan nilai tertentu, jika ditemukan.

DelLastLinkedList(): Menghapus elemen terakhir *linked list* dan mengembalikan *address*-nya.

DelAfterLinkedList(): Menghapus elemen *linked list* setelah elemen tertentu dan mengembalikan *address*-nya.

PrintInfoLinkedList(): Mencetak semua elemen dalam linked list.

NbElmtLinkedList(): Menghitung jumlah elemen dalam *linked list*, mengembalikan 0 jika kosong.

InversLinkedList(): Membalik urutan elemen dalam *linked list* tanpa melakukan alokasi atau dealokasi.

KonkatLinkedList(): Menggabungkan dua *linked list* menjadi linked list baru. Dua *linked list* awal menjadi kosong setelah operasi ini.

ADT Linked List dipilih karena memiliki keunggulan dalam alokasi memori dinamis yang memungkinkan efisiensi penggunaan ruang memori, terutama ketika ukuran data berubah seiring waktu. Linked list juga mendukung penyisipan dan penghapusan elemen secara cepat, karena hanya perlu memperbaiki referensi pointer tanpa perlu melakukan pergeseran data. Struktur data ini diimplementasikan sebagai ADT Linked List dengan nama file header “*linked_list.h*”.

3 Program Utama

Pada program aplikasi PURRMART di Milestone 2, alur program dijalankan dan dimulai dari fungsi-fungsi yang telah dibuat pada Milestone 1. Program ini kemudian akan memerintahkan untuk input beberapa *command*, yaitu *START*, *LOAD*, dan *HELP* sebagai masukan pertama dari aplikasi PURRMART.

Program pada Milestone 2 ini memiliki beberapa *command* setelah **file konfigurasi telah berhasil dibaca**:

1. **PROFILE:** Melihat data diri pengguna, hanya dapat dipanggil saat status pengguna telah *login*.
2. **CART ADD:** Menambahkan barang dengan kuantitas tertentu ke dalam keranjang belanja.
3. **CART REMOVE:** Mengurangi barang sebanyak kuantitas tertentu dari keranjang belanja.
4. **CART SHOW:** Menunjukkan barang-barang yang sudah dimasukkan ke dalam keranjang pengguna.
5. **CART PAY:** Memungkinkan pengguna untuk membeli barang-barang yang sudah dimasukkan ke dalam keranjang.
6. **HISTORY:** Menunjukkan riwayat pembelian dari seorang pengguna.
7. **WISHLIST ADD:** Menambahkan suatu barang ke dalam *wishlist*.
8. **WISHLIST SWAP:** Menukar barang posisi ke-i dengan barang posisi ke-j pada daftar *wishlist*.
9. **WISHLIST REMOVE <i>:** Menghapus barang dengan posisi ke-i dari list *wishlist* pengguna.
10. **WISHLIST REMOVE:** Menghapus barang dari *wishlist* berdasarkan nama barang yang dimasukkan pengguna
11. **WISHLIST CLEAR:** Menghapus semua barang yang terdapat di dalam *wishlist*.
12. **WISHLIST SHOW:** Menunjukkan barang-barang yang sudah dimasukkan ke dalam *wishlist*.

Sebelum mengakses semua fitur tersebut, pengguna dapat memilih *save file* yang ingin mereka gunakan, kemudian pengguna juga dapat menampilkan nama barang yang dijual beserta harganya.

4 Algoritma-Algoritma Menarik

4.1 Algoritma 1

Algoritma yang kami maksud adalah fungsi Lexical, fungsi ini membandingkan dua string secara leksikal. Fungsi ini dapat digunakan dalam algoritma pengurutan dan pencarian, yang termasuk dalam konsep dasar ilmu komputer, algoritma ini memberikan pemahaman perbandingan string seperti pohon pencarian biner.

```
int Lexical(char *str1, char *str2){
    int i = 0;
    while (str1[i] != '\0' && str2[i] != '\0'){
        if (str1[i] < str2[i]) return -1;
        else if (str1[i] > str2[i]) return 1;
        i++;
    }
    if (str1[i] == '\0' && str2[i] != '\0') return -1;
    if (str1[i] != '\0' && str2[i] == '\0') return 1;
    return 0;
}
```

4.2 Algoritma 2

Algoritma yang kami maksud adalah MultiWordWord, fungsi ini mengumpulkan beberapa kata menjadi satu kata panjang, dengan batasan panjang tertentu. Algoritma ini menarik karena menunjukkan bagaimana teks dapat diproses dan dikumpulkan, hal tersebut merupakan bagian penting dari pemrograman teks. Dalam algoritma ini juga dilakukan pengelolaan panjang string dan memastikan tidak melebihi batas yang ditentukan, merupakan contoh manajemen memori yang baik.

```
Word MultiWordWord(){
    Word name;
    name.Length = 0;

    char goods[50] = "";
    int length = 0;
```

```

while (true){
    ADVWORD();
    GoodsWithManyWords(goods, &length, currentWord);

    if (isEndWord()){
        break;
    }
}

if (length > 0 && goods[length - 1] == ' '){
    goods[length - 1] = '\\0';
    length -= 1;
}

if (length > 50){
    length = 50;
}

for (int j = 0; j < length && j < 50; j++){
    name.TabWord[j] = goods[j];
}
name.TabWord[length] = '\\0';
name.Length = length;

return name;
}

```

5 Data Test

5.1 Data Test START

Pada saat program PURRMART pertama kali dijalankan, pengguna dapat menggunakan *command* START untuk mengkonfigurasi program PURRMART dengan *file* konfigurasi *default*. Ketika *command* dijalankan, program membaca *file* default.txt yang berisi konfigurasi *default*.

```
>> START

Save file berhasil dibaca. PURRMART berhasil dijalankan.

>> █
```

Gambar 6.1.1 Data Test Start

5.2 Data Test LOAD

User dapat menggunakan *command* LOAD <nama_file> dengan menginput angka yang bersesuaian dengan opsi LOAD (angka 2) untuk diarahkan menuju langkah selanjutnya, yaitu input nama file (tanpa ekstensi .txt). Ketika pengguna sudah menjalankan seluruh prosedur yang diperintahkan, maka aplikasi akan mengambil *state* yang sudah pernah disimpan oleh pengguna sebelumnya atau *default state* aplikasi tersebut. Namun, jika pengguna menginput nama file yang tidak sesuai, maka program akan gagal dijalankan.

```
>> LOAD save1.txt

Save file berhasil dibaca. PURRMART berhasil dijalankan.

>> LOAD save1,txt

Save file tidak ditemukan. PURRMART gagal dijalankan.
```

Gambar 6.2.1 Data Test Load

5.3 Data Test SAVE

Pengguna program dapat memanggil *command* SAVE untuk menyimpan *state* aplikasi PURRMART terbaru ke dalam suatu *file*. Penyimpanan dilakukan pada *folder* tertentu, misalnya folder save.

```
>> SAVE save5.txt

File 'save5.txt' sudah ada. Apakah ingin di-overwrite (Y/N)? Y
Data berhasil disimpan ke ../save/save5.txt
```

```
>> SAVE save6.txt

Data berhasil disimpan ke ../save/save6.txt
```

Gambar 6.3.1 Data Test Save

5.4 Data Test STORE LIST

Pengguna dapat memanggil *command* STORE LIST untuk melihat barang-barang apa saja yang ada di dalam toko beserta harganya. Setiap barang harus memiliki nama yang *unique* dan berbeda dari barang lainnya.

```
>> STORE LIST

ISI TOKO:
=====
| No | Nama Barang | Harga |
=====
| 1 | AK47 | 10 |
| 2 | Lalabu | 20 |
| 3 | Ayam Goreng Crisbar | 20 |
| 4 | Meong | 500 |
=====

>> STORE LIST

TOKO KOSONG!
```

Gambar 6.4.1 Data Test Store List

5.5 Data Test PROFILE

Pengguna dapat memanggil *command* PROFILE untuk melihat data diri. Pengguna hanya dapat melakukan ini saat status pengguna telah login ke dalam aplikasi.

```
>> PROFILE

+-----+
|              Profil Pengguna              |
+-----+
| Nama   : RANKER                          |
| Saldo  : 0                               |
+-----+
```

Gambar 6.5.1 Data Test Profile

5.6 Data Test CART ADD <nama> <n>

Pengguna dapat memanggil *command* CART ADD <nama> <n> untuk menambahkan barang dengan kuantitas tertentu ke dalam keranjang belanja. Apabila yang ditambahkan tidak ada pada STORE, maka sistem akan menyatakan bahwa perintah invalid dan pengguna akan kembali ke menu utama.

```
>> CART ADD AK47 10
```

Berhasil menambahkan 10 AK47 ke keranjang belanja!

```
>> CART ADD AK48 10
```

Barang tidak ada di toko!

Gambar 6.6.1 Data Test Cart Add

5.7 Data Test CART REMOVE <nama> <n>

Pengguna dapat memanggil *command* CART REMOVE <nama> <n> untuk mengurangi barang dengan kuantitas tertentu dari keranjang belanja. Apabila kuantitas pada keranjang belanja lebih sedikit dari N atau nama barang tidak ada dalam keranjang maka perintah akan gagal dan pengguna akan kembali ke menu utama.

```
>> CART REMOVE AK47 5

Berhasil mengurangi 5 AK47 dari keranjang belanja!

>> CART REMOVE AK47 50

Tidak berhasil mengurangi, hanya terdapat 5 AK47 pada keranjang!

>> CART REMOVE AK48 50

Barang tidak ada di toko!
```

Gambar 6.7.1 Data Test Cart Remove

5.8 Data Test CART SHOW

Pengguna dapat memanggil *command* CART SHOW untuk menunjukkan barang-barang yang sudah dimasukkan ke dalam keranjang beserta total biaya yang harus dikeluarkan untuk membelinya. Apabila tidak ada apapun dalam keranjang, maka sistem akan menyatakan bahwa keranjang pengguna kosong.

```
>> CART SHOW

=====
| Kuantitas | Nama Barang | Total |
=====
| 5         | AK47       | 50    |
=====

Total biaya yang harus dikeluarkan adalah 50.

>> CART SHOW

Keranjang kamu kosong!
```

Gambar 6.8.1 Data Test Cart Show

5.9 Data Test CART PAY

Pengguna dapat memanggil *command* CART PAY untuk membeli barang-barang yang sudah dimasukan ke dalam keranjang. Pastikan bahwa pengguna memiliki saldo yang mencukupi untuk membeli semua barang di dalam keranjang. Proses pembelian akan mengurangi saldo pengguna dan mencatat transaksi ke dalam riwayat pembelian. Barang yang dicatat dalam riwayat pembelian adalah barang dengan total harga tertinggi, yang

dihitung dari perkalian antara harga satuan barang dan kuantitasnya. Jika terdapat beberapa barang dengan total harga yang sama, maka barang yang dipilih adalah yang memiliki urutan leksikal lebih besar. Selain itu, total harga keseluruhan dari pembelian tersebut juga akan disertakan dalam riwayat.

```
>> CART PAY
```

```
Kamu akan membeli barang-barang berikut:
```

Kuantitas	Nama Barang	Total
1	AK47	10

```
Total biaya yang harus dikeluarkan adalah 10, apakah jadi dibeli? (Ya/Tidak): Ya
Selamat kamu telah membeli barang-barang tersebut!
```

```
>> CART PAY
```

```
Kamu akan membeli barang-barang berikut:
```

Kuantitas	Nama Barang	Total
10	AK47	100

```
Total biaya yang harus dikeluarkan adalah 100, apakah jadi dibeli? (Ya/Tidak): Ya
Uang kamu hanya 0, tidak cukup untuk membeli keranjang!
```

```
>> CART PAY
```

```
Kamu akan membeli barang-barang berikut:
```

Kuantitas	Nama Barang	Total
10	AK47	100

```
Total biaya yang harus dikeluarkan adalah 100, apakah jadi dibeli? (Ya/Tidak): Tidak
Pembelian dibatalkan
```



```
>> CART PAY

Kamu akan membeli barang-barang berikut:
=====
| Kuantitas | Nama Barang | Total |
=====
| 10        | AK47        | 100   |
=====

Total biaya yang harus dikeluarkan adalah 100, apakah jadi dibeli? (Ya/Tidak): Nah
Input tidak valid. Kembali ke menu utama.
```

Gambar 6.9.1 Data Test Cart Pay

5.10 Data Test HISTORY <n>

Pengguna dapat memanggil *command* HISTORY <n> untuk menunjukkan riwayat pembelian seorang pengguna. Jika N melebihi jumlah riwayat pembelian yang ada, maka seluruh riwayat pembelian akan ditampilkan. Urutan penunjukan adalah dari yang paling baru ke paling tua.

```
>> HISTORY 4

Riwayat Pembelian:
=====
| No  | Nama Barang | Harga Total |
=====
| 1   | AK47        | 40          |
| 2   | AK47        | 100         |
| 3   | AK47        | 10          |
=====

>> HISTORY 1000

HISTORY KOSONG!
```

Gambar 6.10.1 Data Test History

5.11 Data Test WISHLIST ADD

Pengguna dapat memanggil *command* WISHLIST ADD untuk menambahkan suatu barang ke *wishlist*. Apabila sudah ada di *wishlist*, maka fungsi akan menyatakan bahwa barang sudah ada di *wishlist*. Jika tidak ada barang dengan nama barang yang dimasukkan pengguna, maka fungsi akan menyatakan bahwa tidak ada barang tersebut.

```
>> WISHLIST ADD
```

```
Masukkan nama barang: Meong  
Berhasil menambahkan Meong ke wishlist!
```

```
>> WISHLIST ADD
```

```
Masukkan nama barang: Ayam Geprak Bakar Crispy Besthal  
Tidak ada barang dengan nama Ayam Geprak Bakar Crispy Besthal!
```

Gambar 6.11.1 Data Test Wishlist Add

5.12 Data Test WISHLIST SWAP <i> <j>

Pengguna dapat memanggil *command* WISHLIST SWAP <i> <j> 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. Apabila hanya terdapat satu barang, maka posisi barang tidak dapat ditukar.

```
>> WISHLIST SWAP 2 3
```

```
Berhasil menukar posisi barang pada indeks 2 dengan indeks 3.
```

```
>> WISHLIST SWAP 2 5
```

```
Gagal menukar posisi! Indeks tidak valid.
```

Gambar 6.12.1 Data Test Wishlist Swap

5.13 Data Test WISHLIST REMOVE <i>

Pengguna dapat memanggil *command* WISHLIST REMOVE <i> untuk menghapus barang dengan posisi ke-i dari *wishlist*. Apabila pengguna memasukkan urutan barang yang tidak ada, maka penghapusan barang akan gagal dilakukan, sama halnya ketika tidak ada barang dalam *wishlist*, maka penghapusan akan gagal untuk dilakukan. Hal ini berlaku juga ketika 'i' bukan merupakan angka, maka fungsi akan menyatakan bahwa *command* tidak valid.

```
>> WISHLIST REMOVE 2

Berhasil menghapus barang posisi ke-2 dari wishlist!

>> WISHLIST REMOVE 5

Penghapusan barang WISHLIST gagal dilakukan, Barang ke-5 tidak ada di WISHLIST!

>> WISHLIST REMOVE Y

Penghapusan barang WISHLIST gagal dilakukan, Posisi tidak valid!

>> WISHLIST REMOVE 4

Penghapusan barang WISHLIST gagal dilakukan, WISHLIST kosong!
```

Gambar 6.13.1 Data Test Wishlist Remove <i>

5.14 Data Test WISHLIST REMOVE

Pengguna dapat memanggil *command* WISHLIST REMOVE untuk menghapus barang dari *wishlist* berdasarkan nama barang yang dimasukkan pengguna. Apabila nama barang tidak ada dalam *wishlist*, maka penghapusan barang akan gagal untuk dilakukan.

```
>> WISHLIST REMOVE

Masukkan nama barang: Meong
Meong berhasil dihapus dari WISHLIST!

>> WISHLIST REMOVE

Masukkan nama barang: Ayam Goreng
Penghapusan barang WISHLIST gagal dilakukan, Ayam Goreng tidak ada di WISHLIST!
```

Gambar 6.13.1 Data Test Wishlist Remove <i>

5.15 Data Test WISHLIST CLEAR

Pengguna dapat memanggil *command* WISHLIST CLEAR untuk menghapus semua barang yang terdapat di dalam WISHLIST.

```
>> WISHLIST CLEAR

wishlist telah dikosongkan.
```

Gambar 6.15.1 Data Test Wishlist Clear

5.16 Data Test WISHLIST SHOW

Pengguna dapat memanggil *command* WISHLIST SHOW untuk menunjukkan barang-barang yang sudah dimasukkan ke dalam *wishlist*. Apabila tidak ada barang apapun dalam *wishlist*, maka fungsi akan menyatakan bahwa *wishlist* kosong.

```
>> WISHLIST SHOW

ISI WISHLIST:
+-----+
|                WISHLIST                |
+-----+
| AK47                                     |
| Meong                                   |
| Lalabu                                 |
+-----+
```

```
>> WISHLIST SHOW

WISHLIST KOSONG!
```

Gambar 6.16.1 Data Test Wishlist Show

6 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	START	Memeriksa apakah <i>file config</i> default berhasil dibaca	<ul style="list-style-type: none"> Melakukan <i>compile</i> semua <i>file</i> yang berkaitan dengan <i>main.c</i> pada terminal Menginput <i>command</i> START pada saat pertama kali memulai program PurrMart 	START	Program mengeluarkan "File konfigurasi aplikasi berhasil dibaca. PURRMART berhasil dijalankan."	Program mengeluarkan "File konfigurasi aplikasi berhasil dibaca. PURRMART berhasil dijalankan."

2	LOAD	Memeriksa apakah <i>file</i> berhasil dibaca	<ul style="list-style-type: none"> - Melakukan <i>compile</i> semua <i>file</i> yang berkaitan dengan <i>main.c</i> pada terminal - Memasukkan <i>command</i> LOAD <filename> 	LOAD <filename>	Kondisi 1: Program mengeluarkan "Save file tidak ditemukan. PURRMART gagal dijalankan." Kondisi 2: "Save file berhasil dibaca. PURRMART berhasil dijalankan."	Kondisi 1: Program mengeluarkan "Save file tidak ditemukan. PURRMART gagal dijalankan." Kondisi 2: "Save file berhasil dibaca. PURRMART berhasil dijalankan."
3	HELP	Memunculkan daftar <i>command</i> yang dapat digunakan oleh pengguna	<ul style="list-style-type: none"> - Memasukkan <i>command</i> HELP 	HELP	Program mengeluarkan menu Help	Program mengeluarkan menu Help
4	PROFILE	Menampilkan data diri pengguna berupa nama dan saldo pengguna	Pengguna sudah <i>compile</i> , loaded, dan login. Memasukkan <i>command</i> "PROFILE"	PROFILE	Program mengeluarkan pesan "Nama : '....' Saldo: '....'"	Program mengeluarkan pesan "Nama : '....' Saldo: '....'"
5	CART ADD	Menambahkan barang dengan kuantitas tertentu ke dalam keranjang belanja	Pengguna sudah <i>compile</i> , loaded, dan login. Memasukkan <i>command</i> "CART ADD <nama> <n>"	CART ADD <nama> <n>	Kondisi 1: Program mengeluarkan "Berhasil menambahkan '...' '...' ke keranjang belanja!" Kondisi 2: "Barang tidak ada di toko!"	Kondisi 1: Program mengeluarkan "Berhasil menambahkan '...' '...' ke keranjang belanja!" Kondisi 2: "Barang tidak ada di toko!"
6	CART REMOVE	Mengurangi barang sejumlah kuantitas tertentu dari keranjang belanja	Pengguna sudah <i>compile</i> , loaded, dan login. Memasukkan <i>command</i> "CART REMOVE"	CART REMOVE <nama> <n>	Program mengeluarkan pesan "Berhasil mengurangi '...' '...' dari keranjang belanja!"	Program mengeluarkan pesan "Berhasil mengurangi '...' '...' dari keranjang belanja!"
7	CART SHOW	Menunjukkan barang-barang yang sudah dimasukkan ke dalam keranjang	Pengguna sudah <i>compile</i> , loaded, dan login. Memasukkan <i>command</i> "CART SHOW"	CART SHOW	Program mengeluarkan pesan "Berikut adalah isi keranjangmu. Kuantitas '...' Nama '...' Total '...' Total biaya yang harus dikeluarkan adalah '...'"	Program mengeluarkan pesan "Berikut adalah isi keranjangmu. Kuantitas '...' Nama '...' Total '...' Total biaya yang harus dikeluarkan adalah '...'"
8	CART PAY	Membeli barang-barang yang sudah dimasukkan ke dalam keranjang	Pengguna sudah <i>compile</i> , loaded, dan login. Memasukkan <i>command</i> "CART PAY"	CART PAY	Program mengeluarkan pesan "Kamu akan membeli barang-barang berikut. Kuantitas '...' Nama '...' Total '...'"	Program mengeluarkan pesan "Kamu akan membeli barang-barang berikut. Kuantitas '...' Nama '...' Total '...'"

					<p>‘...’ Total biaya yang harus dikeluarkan adalah ‘...’, apakah jadi dibeli? (Ya/Tidak): ‘...’ Selamat kamu telah membeli barang-barang tersebut!”</p>	<p>‘...’ Total biaya yang harus dikeluarkan adalah ‘...’, apakah jadi dibeli? (Ya/Tidak): ‘...’ Selamat kamu telah membeli barang-barang tersebut!”</p> <p>Jika keranjang kosong, maka program mengeluarkan “Keranjang kamu kosong!”</p> <p>Jika tidak ingin membeli, maka program mengeluarkan “Pembelian dibatalkan”</p>
9	HISTORY	Menunjukkan riwayat pembelian seorang pengguna	Pengguna sudah <i>compile</i> , loaded, dan login. Memasukkan <i>command</i> “HISTORY”	HISTORY	Program mengeluarkan pesan “Riwayat pembelian barang: “...”	Program mengeluarkan pesan “Riwayat pembelian barang: “...”
10	WISHLIST ADD	Menambahkan barang ke dalam <i>wishlist</i>	Pengguna sudah <i>compile</i> , loaded, dan login. Memasukkan <i>command</i> “WISHLIST ADD”	WISHLIST ADD	Program mengeluarkan pesan “Masukkan nama barang: ‘...’ Berhasil menambahkan ‘...’ ke wishlist!”	Program mengeluarkan pesan “Masukkan nama barang: ‘...’ Berhasil menambahkan ‘...’ ke wishlist!”
11	WISHLIST SWAP	Menukar barang posisi ke-i dengan barang posisi ke-j pada <i>wishlist</i>	Pengguna sudah <i>compile</i> , loaded, dan login. Memasukkan <i>command</i> “WISHLIST SWAP <i> <j>”	WISHLIST SWAP <i> <j>	Program mengeluarkan pesan “Berhasil menukar posisi ‘...’ dengan ‘...’ pada wishlist!”	Program mengeluarkan pesan “Berhasil menukar posisi ‘...’ dengan ‘...’ pada wishlist!”
12	WISHLIST REMOVE	Menghapus barang dengan posisi ke-i dari <i>wishlist</i>	Pengguna sudah <i>compile</i> , loaded, dan login. Memasukkan <i>command</i> “WISHLIST REMOVE <i>”	WISHLIST REMOVE <i>	Program mengeluarkan pesan “Berhasil menghapus barang posisi ke-’...’ dari wishlist!”	Program mengeluarkan pesan “Berhasil menghapus barang posisi ke-’...’ dari wishlist!”
						Jika tidak barang di dalam list, maka program

						<p>mengeluarkan "Penghapusan barang WISHLIST gagal dilakukan, WISHLIST kosong!"</p> <p>Jika dibawah 1, maka program mengeluarkan "Penghapusan barang WISHLIST gagal dilakukan, Posisi tidak valid!"</p>
13	WISHLIST REMOVE	Menghapus barang dari <i>wishlist</i> berdasarkan nama barang yang dimasukkan pengguna	Pengguna sudah <i>compile</i> , loaded, dan login. Memasukkan <i>command</i> "WISHLIST REMOVE"	WISHLIST REMOVE	Program mengeluarkan pesan "Masukkan nama barang yang akan dihapus : '...'" '...' berhasil dihapus dari WISHLIST!"	<p>Program mengeluarkan pesan "Masukkan nama barang yang akan dihapus : '...'" '...' berhasil dihapus dari WISHLIST!"</p> <p>Jika tidak barang di dalam list, maka program mengeluarkan "Penghapusan barang WISHLIST gagal dilakukan, WISHLIST kosong!"</p>
14	WISHLIST CLEAR	Menghapus semua barang yang terdapat di dalam <i>wishlist</i> .	Pengguna sudah <i>compile</i> , loaded, dan login. Memasukkan <i>command</i> "WISHLIST CLEAR"	WISHLIST CLEAR	Program mengeluarkan pesan "Wishlist telah dikosongkan" dan tidak ada lagi barang di dalam <i>wishlist</i>	<p>Program mengeluarkan pesan "Wishlist telah dikosongkan" dan tidak ada lagi barang di dalam <i>wishlist</i></p> <p>Jika <i>wishlist</i> kosong, maka program mengeluarkan "Wishlist Kosong!"</p>
15	WISHLIST SHOW	Menunjukkan barang-barang yang sudah dimasukkan ke dalam <i>wishlist</i> .	Pengguna sudah <i>compile</i> , loaded, dan login. Memasukkan <i>command</i> "WISHLIST SHOW"	WISHLIST SHOW	Program mengeluarkan pesan "Berikut adalah isi wishlist-mu: '...'"	Program mengeluarkan pesan "Berikut adalah isi wishlist-mu: '...'"

7 Pembagian Kerja dalam Kelompok

Nama Anggota Kelompok – NIM	Pembagian Kerja
Syasya Umaira – 18221026	<ul style="list-style-type: none">- Membuat wishlist remove, wishlist remove ke-i- Membuat algoritma menarik
Kezia Karen Cahyadi - 18222041	<ul style="list-style-type: none">- Membuat wishlist clear, deteksi kebocoran biologis
Raka Adhitya Nugraha – 18223013	<ul style="list-style-type: none">- Membuat ADT LinkedList dan Map- Membuat ADT Driver- Membuat Profile, CartAdd, CartShow- Menuliskan laporan bagian Ringkasan, Struktur Data (ADT), Data Test dan Pembagian Tugas
Michelle Hamdani – 18223037	<ul style="list-style-type: none">- Membuat CartRemove, CartPay, History- Membuat test script
Daffa Athalla Rajasa – 18223053	<ul style="list-style-type: none">- Membuat wishlist add dan wishlist swap- Menulis laporan bagian lampiran (<i>log activity</i> kelompok)- Manajemen Asistensi (Kontak Asisten dan Notulensi)
Atharizza Muhammad Athaya – 18223079	<ul style="list-style-type: none">- Mengintegrasikan Kode ke Main- Mengupgrade Save dan Load- Mengupgrade ADT Custom- Membuat ADT Stack

8 Lampiran

8.1 Deskripsi Tugas Besar

Buatlah sebuah aplikasi simulasi berbasis CLI (command-line interface). Sistem ini dibuat dalam **bahasa C** dengan menggunakan **struktur data yang sudah kalian pelajari** di mata kuliah ini. Kalian boleh menggunakan (atau memodifikasi) struktur data yang sudah kalian buat untuk

praktikum pada tugas besar ini. Daftar ADT yang wajib digunakan dapat dilihat pada bagian Daftar ADT. Library yang boleh digunakan hanya **stdio.h**, **stdlib.h**, **time.h**, dan **math.h**.

System Mechanic

1. About the System

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

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

2. Menu Program

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

3. Command

Pengguna dapat memasukkan *command-command* berikut. Seluruh *command* hanya berlaku pada menu utama.

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. WISHLIST ADD

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

g. 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.

h. WISHLIST REMOVE <i>

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

i. WISHLIST REMOVE

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

j. WISHLIST CLEAR

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

k. WISHLIST SHOW

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

4. 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.




5. Bonus

8.2 Notulen Rapat

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

No. Kelompok/Kelas : 11 / K01
Nama Kelompok : *Purrmart*
Anggota Kelompok (Nama/NIM) : 1. Syasya Umaira (18221026)
2. Kezia Caren Cahyadi (18222046)
3. Raka Adhitya Nugraha (18223013)
4. Michelle Hamdani (18223037)
5. Daffa Athalla Rajasa (18223053)
6. Atharizza Muhammad Athaya (18223079)
Asisten Pembimbing : Jonathan Arthurito Aldi Sinaga

Asistensi I

Tanggal : 22 November 2024 21.00 - selesai	Catatan Asistensi:
Tempat : Google Meet (online)	
Kehadiran Anggota Kelompok: 1. Syasya Umaira (18221026)  2. Kezia Karen Cahyadi (18222041) 3. Raka Adhitya Nugraha (18223013)  4. Michelle Hamdani (18223037) 	<p>Pertanyaan 1: Bonus spesifikasi riwayat maksimal, stack yang bagus kayak gimana? Cuma masih gabisa ngerun padahal sudah bener, kayak rte loading lama dan berhenti. Jawaban: Terserah kalian sih mau kayak gimana, stack of linkedlist sudah bener</p> <p>Pertanyaan 2: Untuk driver diupdate lagi berarti untuk ADT baru? Jawaban: Iya diubah, kecuali masih sama kaya milestone 1, gak perlu diubah-ubah</p> <p>Pertanyaan 3: Buat load pasti ngikutin format kan ya? Gamungkin aneh-aneh filenya? Jawaban: File sudah pasti benar, dan tidak usah dipikirkan</p> <p>Pertanyaan 4: Buat wishlist add itu gimana kak? Jawaban: Buat wishlist add nge-pas-in list dinamis yang ada aja</p> <p>Pertanyaan 5: Mau tanya kak, buat algoritma menarik itu fungsi yang kita buat sendiri buat bantu programnya atau gimana ya? Jawaban: Isinya tentang algoritma-algoritma yang dirasa menarik saat kamu nyelesain tugas besar ini, misalkan di ADT nya, atau di console nya.</p>

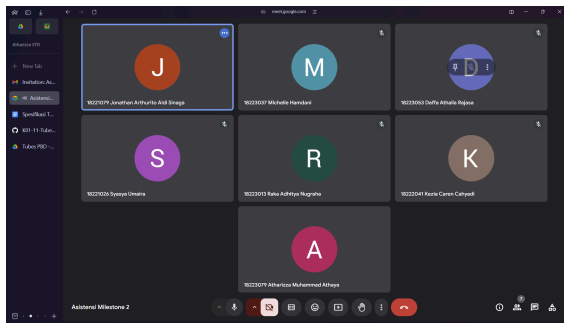
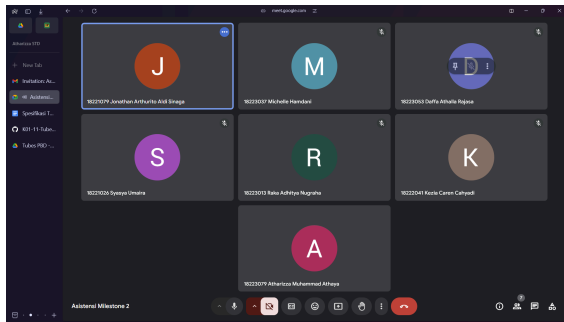
5. Daffa Athalla Rajasa (18223053)

[Handwritten signature of Daffa Athalla Rajasa]

6. Atharizza Muhammad Athaya (18223079)

[Handwritten signature of Atharizza Muhammad Athaya]

Dokumentasi:



Tanda Tangan Asisten:

[Handwritten signature of the Assistant]

8.3 Log Activity Anggota Kelompok

Tanggal	Nama (NIM)	Aktivitas
10 Desember 2024	18223079 Atharizza Muhammad Athaya	Update QuantumQWordl3
11 Desember 2024	18223079 Atharizza Muhammad Athaya	add stack ADT edit load
12 Desember 2024	18223079 Atharizza Muhammad Athaya	update custom.c and stack.h
16 Desember 2024	18223079 Atharizza Muhammad Athaya	repair custom
17 Desember 2024	18223013 Raka Adhitya Nugraha	Merge remote-tracking branch 'refs/remotes/origin/main' feat: Add ADT Map feat: Add ADT Linked List feat: Add Cart Add, Remove, Show
	18223037 Michelle Hamdani	Add files via upload
	18223079 Atharizza Muhammad Athaya	update Makefile, map, and cartadd move cartshow to console
18 Desember 2024	18223013 Raka Adhitya Nugraha	feat: Add Profile, History (not fix), Repair ADT Stack
	18223037 Michelle Hamdani	Add files via upload Create cart_remove.h Update store_list.c Create cart_pay.h Create history.h Create riwayat_maksimal.h Add files via upload Update cart_remove.h Update cart_pay.h Update history.h Update riwayat_maksimal.h
	18223079 Atharizza Muhammad Athaya	move cartremove to console integrating some functions into console update user Merge branch 'main' of

		https://github.com/ITBAtharizza/K01-1-1-TubesADT fix multiple declaration update Load update main
	18221026 Syasya Umaira	add wishlist remove index i and remove update wishlist remove index i and remove
	18223053 Daffa Athalla Rajasa	feat wishlist_add.h Refactor wishlist_add.h Merge branch 'main' of https://github.com/ITBAtharizza/K01-1-1-TubesADT
19 Desember 2024	18223013 Raka Adhitya Nugraha	feat: Add Profile to Console and Main
20 Desember 2024	18223079 Atharizza Muhammad Athaya	Update dumpUser repair Save and FlipStack Update save logic
	18222041 Kezia Caren Cahyadi	Create wishlist_clear.c Create wishlist_clear.h Update wishlist_clear.c Update wishlist_clear.h Create store_list_gacor.c Create biological_leakage.c Create store_list_gacor.h Create biological_leakage.h Update biological_leakage.c
	18223053 Daffa Athalla Rajasa	Feat wishlist_add.c Feat wishlist_swap.h Feat wishlist_swap.c
	18223013 Raka Adhitya Nugraha	feat: Add ADT Driver
20 Desember 2024	18223079 Atharizza Muhammad Athaya	fix wishlist add and swap add dumpuser and reversedump update maxEl fix wishlist add

		fix bug finishing touch
	18223013 Raka Adhitya Nugraha	Add Document