

LAPORAN TUGAS BESAR


IF2111 Algoritma dan Struktur Data STI

PurrMart

Dipersiapkan oleh:
K1-03

Wisa Ahmaduta Dinutama	18223003
Carissa Zahrani Putri	18221093
Catherine Alicia N	18223069
Muhammad Aqmar Fayyaz Zakaria	18223043
Hugo Benedicto Tanidi	18221131

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

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		<i>IF2111-TB-01-03</i>		
		<i>Revisi</i>	<i>1</i>	<i>25-11-2024</i>

Daftar Isi

1	Ringkasan	3
2	Penjelasan Tambahan Spesifikasi Tugas	6
	2.1 Quantum WORDL3	6
	2.2 Bioweapon	6
3	Struktur Data (ADT)	6
	3.1 Struktur Data Array Items	6
	3.2 Struktur Data Array User	7
	3.3 Struktur Data Mesin Karakter	8
	3.4 Struktur Data Mesin Kata	8
	3.5 Struktur Data Queue	9
	3.6 Struktur Data Stack	10
4	Program Utama	11
5	Algoritma-Algoritma Menarik	12
	5.1 Konversi DNA ke RNA	12
6	Data Test	12
	6.1 Data Test START	12
	6.2 Data Test LOAD	13
	6.3 Data Test HELP	14
	6.4 Data Test REGISTER	16
	6.5 Data Test LOGIN	17
	6.6 Data Test WORK	19
	6.7 Data Test WORK CHALLENGE	20
	6.7.1 Data Test Tebak Angka	21
	6.7.2 Data Test WORDL3	23
	6.7.3 Data Test Quantum WORDL3	25
	6.8 Data Test BIOWEAPON	26
	6.9 Data Test STORE LIST	27
	6.10 Data Test STORE REQUEST	28
	6.11 Data Test STORE SUPPLY	29
	6.12 Data Test STORE REMOVE	30
	6.13 Data Test LOGOUT	30
	6.14 Data Test EXIT	31
	6.15 Data Test SAVE	32
7	Test Script	33

8	Pembagian Kerja dalam Kelompok	35
9	Lampiran	37
9.1	Deskripsi Tugas Besar	37
1.	About the System	37
2.	Main Menu	37
3.	Command	38
a.	START	38
b.	LOAD <filename>	38
c.	LOGIN	38
d.	LOGOUT	38
e.	REGISTER	38
f.	WORK	38
g.	WORK CHALLENGE	39
h.	STORE LIST	39
i.	STORE REQUEST	39
j.	STORE SUPPLY	40
k.	STORE REMOVE	40
l.	HELP	40
m.	SAVE <filename>	40
n.	QUIT	40
9.2	Notulen Rapat	40
9.3.1	Notulensi Asistensi 1	40
9.3	Log Activity Anggota Kelompok	43

1 Ringkasan

PurrMart merupakan suatu aplikasi yang mensimulasikan aktivitas beli barang pada *e-commerce* yang berbasis *Command-Line Interface (CLI)* yang memungkinkan pengguna dapat menikmati berbagai fitur terkait pembelian barang. 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*, dan bekerja untuk menghasilkan uang. Perubahan yang dilakukan terhadap aplikasi, seperti penggunaan fitur, bisa disimpan / *save* pada sebuah *file* dengan *type* *txt*. Aplikasi ini dirancang menggunakan bahasa pemrograman C dan mengimplementasikan struktur data yang relevan untuk memenuhi kebutuhan fitur / layanan yang diperlukan.

Aplikasi PurrMart memiliki alur kerja dan interaksi pengguna. Pertama, **keberjalanan aplikasi** yang dimulai dengan tampilan *welcome menu* dari PurrMart yang dapat menerima *command* dengan opsi START (1), LOAD (2), EXIT (3), dan HELP (4). Dengan memilih START, aplikasi membaca file konfigurasi *default* yang berisi jumlah barang, harga barang beserta namanya, jumlah user, dan uang-nama-password dari user. Kemudian, LOAD memungkinkan pengguna untuk memuat *save file* sebelumnya dari *folder saves* untuk melanjutkan sesi sebelumnya. *Command* EXIT memungkinkan pengguna untuk keluar dari sesi aplikasi, dengan atau tanpa menyimpan data sesi terkini, sesuai dengan pilihan pengguna.

Kedua, terkait **pembuatan atau pengaksesan akun**. Setelah aplikasi sudah dijalankan, user akan diarahkan ke *login menu* yang dapat menerima *command* dengan opsi LOGIN, LOGOUT, REGISTER, HELP, dan EXIT. Dengan memilih LOGIN, harus dipastikan bahwa user sudah pernah melakukan REGISTER atau data username dan password telah tersimpan dalam file konfigurasi yang dimuat agar username dan password dapat terlacak. PurrMart akan terus meminta input username dan password sampai benar dengan keterangan username dan password harus sesuai, user sudah pernah mendaftarkan akunnya, dan sedang tidak ada user yang login. Kemudian, REGISTER berguna untuk mendaftarkan akun baru ke dalam sistem PurrMart. Username yang didaftarkan harus unik, PurrMart tidak akan menerima pendaftaran akun jika username yang diinput sudah ada. Setelah pengguna berhasil *login* ke PurrMart, akan dilanjutkan dengan *main menu*. Pada *main menu*, pengguna dapat memberi *command* BACK untuk balik ke *login menu* dengan pengguna yang masih sama.

Ketiga, terkait **navigasi daftar work**. Pengguna dapat menggunakan *command* WORK (1) untuk melihat pekerjaan yang tersedia, setiap pekerjaan memiliki waktu tunggu yang berbeda-beda dan dengan nominal pendapatan yang berbeda-beda juga. Pengguna juga dapat memilih pekerjaan yang ingin dipilih. Pengguna tidak dapat memilih pekerjaan selama pengguna sedang bekerja.

Keempat, **permainan work challenge**. Pengguna memasukkan *command* WORK CHALLENGE (2), kemudian memilih daftar *challenge* yang akan dimainkan. Pengguna membutuhkan uang dengan jumlah tertentu untuk memainkan *challenge*. Uang yang dibayarkan untuk bermain *challenge* tidak akan dikembalikan, meskipun pemain kalah dalam permainan. Permainan yang dapat dimainkan berupa tebak angka, WORDL3 dan QUANTUM WORDL3. Setiap kali pengguna berhasil memenangkan permainan, maka uang dari akun pengguna akan bertambah.

Kelima, **navigasi daftar barang pada toko.** Pengguna dapat menggunakan *command* STORE LIST (3) untuk melihat daftar barang-barang yang tersedia di toko. Jika belum ada barang di toko, maka dimunculkan tulisan TOKO KOSONG.

Keenam, **melakukan request barang untuk dimasukkan ke toko.** Pengguna dapat meminta penambahan barang baru ke dalam toko dengan menggunakan *command* STORE REQUEST (4), tetapi barang tersebut akan dimasukkan ke dalam antrian terlebih dahulu. Sistem tidak akan memasukan permintaan penambahan barang baru kalau barang tersebut sudah ada di toko atau di antrian.

Ketujuh, terkait **memasukan barang ke toko.** *Command* STORE SUPPLY (5) memungkinkan pengguna untuk menentukan apakah barang ini ingin dimasukkan atau tidak ke dalam toko. Terdapat 3 pilihan input, terima, tunda, dan tolak. Jika diterima, maka program akan meminta harga dari barang, lalu barang dimasukkan ke toko dan keluar dari antrian. Jika ditunda, maka barang akan dikembalikan masuk ke dalam antrian. Jika ditolak, maka barang akan dihapus dari antrian dan tidak masuk ke dalam toko. Jika dimasukkan input selain ketiga pilihan tersebut, maka pengguna akan diarahkan kembali ke *main menu*.

Kedelapan, terkait **penghapusan barang di dalam toko.** *Command* STORE REMOVE (6) memungkinkan pengguna untuk menghapus barang yang ada di toko. Sistem tidak akan melakukan perintah kalau barang yang ingin dihapus tidak ada di list barang toko.

Kesembilan, terkait **bantuan informasi aplikasi.** Pengguna dapat memasukkan *command* HELP di beberapa menu, *welcome menu*, *login menu*, dan *main menu*. HELP yang akan ditampilkan akan berbeda-beda setiap menu, sesuai dengan *command* yang memungkinkan untuk diterima di menu tersebut.

Kesepuluh, terkait **simpan status dan keluar dari aplikasi.** *Command* SAVE digunakan untuk menyimpan status atau *state* aplikasi ke dalam *file save*, memungkinkan pengguna untuk mengaksesnya di sesi selanjutnya. *Command* EXIT memungkinkan pengguna untuk keluar dari sesi aplikasi, dengan atau tanpa menyimpan data sesi terkini, sesuai dengan pilihan pengguna.

2 Penjelasan Tambahan Spesifikasi Tugas

Dalam pengembangan PurrMart terdapat tambahan fitur sebagai berikut

2.1 *Quantum WORDL3*

Merupakan fitur work challenge ekstra yang serupa dengan WORDL3, tetapi pemain harus menebak empat kata dalam sembilan percobaan. Setiap input dan kata target merupakan kata lima huruf yang dipilih secara acak dari daftar kata yang dibentuk. Program ini mengimplementasikan ADT Mesin Karakter dan Mesin Kata dalam pembacaan input dari pengguna. Setiap kali tebakan diberikan, program akan membandingkan terhadap keempat kata target sekaligus mengecek kesesuaian posisi huruf. Huruf yang salah diberi tanda %, huruf yang benar tetapi memiliki kesalahan posisi diberi tanda *, dan huruf yang telah benar serta berada di posisi yang tepat tidak memiliki tanda apapun. Permainan ini juga memastikan tidak ada kata target yang dipilih lebih dari sekali dari daftar kata yang ada. Permainan berakhir ketika semua kata berhasil ditebak atau ketika pemain kehabisan kesempatan.

2.2 *Bioweapon*

Merupakan program untuk memproses dan memvalidasi “senjata biologis” melalui analisis sekuens DNA. Program menerima tiga input berupa nama senjata, sekuens DNA (dalam huruf kapital seluruhnya), dan kode rahasia. Pemrosesan input menggunakan ADT Mesin Karakter dan Mesin Kata. Program mengonversi DNA menjadi RNA dengan pemetaan A menjadi U, T menjadi A, G menjadi C, dan C menjadi G. Selanjutnya program mengubah RNA menjadi sekuens protein menggunakan berdasarkan tabel kodon-asam amino. Program kemudian mencari kode rahasia dalam sekuens protein yang dihasilkan untuk menentukan validitas senjata.

3 Struktur Data (ADT)

Dalam program ini, kami menggunakan beberapa ADT, yaitu ADT arrayitems, ADT arrayuser, ADT mesinkarakter, ADT mesinkata, ADT queue, dan ADT stack.

3.1 *Struktur Data Array Items*

Pada pengerjaan program PurrMart, kami menggunakan struktur data *ListofItems* (*struct ListofItems*). Struktur data ini merupakan struktur yang merepresentasikan tipe bentukan sebuah daftar item. Struktur data *ListofItems* yang kami gunakan ini menyimpan beberapa data seperti array penyimpanan data item, kapasitas array, dan jumlah elemen efektif dengan tipe data yang sesuai.

Data array penyimpanan item disimpan dalam pointer array *A* yang bertipe *Item**. Array ini digunakan untuk menyimpan elemen-elemen item yang akan dikelola dalam program. Selanjutnya, data kapasitas disimpan dalam tipe data integer *Capacity*. Data kapasitas ini merepresentasikan kapasitas maksimum dari array saat ini, yang dapat bertambah secara **dinamis** sesuai kebutuhan. Terakhir, data jumlah elemen efektif disimpan dalam tipe data *Neff*. Data ini merepresentasikan jumlah elemen item yang sedang digunakan atau terisi dalam array.

Struktur data *ListofItems* digunakan untuk menyimpan kumpulan item, yang terdiri dari informasi tentang item-item yang akan dimanipulasi dalam program. Struktur data ini nantinya

akan digunakan untuk menambahkan item baru, menghapus item, mencari item, membalikan urutan item, dan berbagai operasi lain yang diperlukan dalam aplikasi.

Kami memilih menggunakan array dinamis karena menurut kami kemungkinan fleksibilitas dalam pengelolaan data yang ukurannya dapat berubah selama runtime. Dengan menggunakan array dinamis, kapasitas array dapat ditingkatkan secara otomatis ketika kapasitas awalnya telah penuh, sehingga dapat menampung lebih banyak item tanpa batas ukuran yang kaku. Hal ini memungkinkan program untuk menangani jumlah data yang besar dan bervariasi tanpa mengalami masalah kapasitas.

Selanjutnya, terdapat beberapa implementasi fungsi seperti *MakeListOfItems*, *IsItemsEmpty*, *LengthListOfItems*, *InsertItemAt*, *InsertLastItem*, *DeleteItemAt*, dan *ReverseListOfItems* digunakan untuk memanipulasi dan menampilkan informasi terkait daftar item. Fungsi-fungsi ini memberikan kemampuan untuk membuat daftar item baru, memeriksa apakah daftar kosong, menambahkan item pada posisi tertentu atau di akhir daftar, menghapus item dari posisi tertentu, membalik urutan item dalam daftar, dan menyalin daftar item. Dengan adanya fungsi-fungsi ini, pengelolaan data menjadi lebih terstruktur dan efisien.

File *arrayitems.h* dan *arrayitems.c* membentuk sebuah ADT “List of Items”, yang berisi struktur data *ListofItems* dan fungsi-fungsi yang beroperasi pada struktur data tersebut. Pendekatan ini memungkinkan penggunaan yang jelas dalam manipulasi data *ListofItems*, menyembunyikan detail implementasi dan menyediakan fungsionalitas terstruktur untuk mengelola informasi terkait item. Ini memberikan modularita, abstraksi, dan memungkinkan penggunaan ulang fungsi-fungsi tersebut di berbagai bagian program. Dengan demikian, pengembangan dan pemeliharaan kode menjadi lebih mudah dan terorganisir.

3.2 Struktur Data Array User

Pada pengerjaan program PurrMart, kami menggunakan struktur data *ListofUsers* (*struct ListofUsers*). Struktur data ini merepresentasikan tabel atau **array statis** yang digunakan untuk menyimpan data pengguna dalam bentuk daftar. Elemen yang disimpan dalam array memiliki tipe *User*, yaitu data khusus yang merepresentasikan informasi dari masing-masing pengguna. Struktur ini digunakan untuk mengelola daftar pengguna dalam aplikasi secara efisien.

Struktur data *ListofUsers* menyimpan informasi dalam tiga atribut utama. Atribut pertama adalah *TI*, yaitu array yang digunakan sebagai tempat penyimpanan elemen-elemen bertipe *User*. Selanjutnya, atribut *Neff* merepresentasikan jumlah elemen efektif yang sedang digunakan dalam array, sehingga hanya elemen dari indeks *IdxMin* hingga *Neff* yang dianggap valid. Selain itu, batas maksimum elemen yang dapat disimpan dalam array ditentukan oleh konstanta *IdxMax*. Dengan struktur ini, elemen dalam daftar pengguna dapat dimanipulasi secara langsung menggunakan indeks yang valid.

ListofUsers mulai berpedan sejak awal program ketika tabel pengguna diinisialisasi menggunakan fungsi *MakeEmpty*, yang mengatur nilai *Neff* menjadi nol untuk menandakan bahwa daftar pengguna kosong. Ketika pengguna baru ditambahkan, fungsi seperti *InsertLastUser* akan menyisipkan elemen baru di akhir tabel, sementara fungsi *DeleteLastUser* digunakan untuk menghapus elemen terakhir dari daftar. Selain itu, fungsi seperti *GetElmt* dan *SetEl* memungkinkan pengaksesan dan pengubahan elemen pada indeks tertentu, sehingga mempermudah manipulasi data pengguna.

File *listofusers.h* dan *listofusers.c* membentuk sebuah ADT *ListofUsers* yang berisi definisi struktur data *ListofUsers* serta implementasi fungsi-fungsi pendukung. Fungsi-fungsi ini mencakup pengelolaan elemen seperti penambahan, penghapusan, penggantian, serta pemeriksaan status kosong atau penuhnya tabel. Dengan pendekatan ini, struktur data *ListofUsers* memberikan solusi yang efisien untuk menyimpan dan mengelola data pengguna secara modular dan terstruktur di dalam aplikasi.

3.3 Struktur Data Mesin Karakter

Pada pengerjaan program PurrMart, kami menggunakan struktur data **Mesin Karakter** untuk membaca dan menulis karakter dari pita atau file. Mesin karakter ini memanfaatkan variable *currentChar* sebagai penanda karakter yang sedang diproses, serta *EOP* (*End of Process*) untuk menandai akhir dari pita. Pita bacaan dikelola menggunakan pointer statis *pita*, yang dapat menunjuk ke file input atau output yang sedang diproses.

Struktur data *Mesin Karakter* berfungsi untuk mengelola aliran karakter yang dibaca dari pita atau file. Mesin ini diawali dengan fungsi *START*, yang mengatur pita bacaan dari standar input (*stdin*), atau fungsi *STARTFILE* yang menginisialisasi pita dari sebuah file tertentu yang ada di folder *saves*. Pada saat mesin membaca karakter, fungsi *ADV* digunakan untuk memajukan pita ke karakter berikutnya. Karakter saat ini dapat diakses melalui fungsi *GetCC*, dan status akhir pita dapat diperiksa menggunakan fungsi *IsEOP*. Selain itu, mesin ini juga mendukung penulisan data ke dalam file menggunakan fungsi seperti *WRITEFILE*, *printChar*, *printInt*, *printNewLine*, dan *printBlank*.

Mesin karakter mulai berperan ketika program memerlukan pembacaan atau penulisan data dari dan ke file. Saat memulai proses membaca file dengan *STARTFILE*, program akan membuka file yang ditunjukkan oleh path tertentu di folder *saves*, dan menginisialisasi mesin dengan karakter pertama dari file. Jika file tidak ditemukan, status pembukaan file akan diberikan melalui parameter keluaran *success*. Untuk proses penulisan, fungsi *WRITEFILE* membuka file tujuan dalam mode tulis dan mempersiapkan pita untuk menerima data karakter atau angka.

File *mesinkarakter.h* dan *mesinkarakter.c* membentuk ADT Mesin Karakter, yang mengelola alur baca-tulis pita dengan fungsi-fungsi yang sesuai. Fungsi seperti *RESETPITA* memastikan posisi pembacaan kembali ke awal file, sementara fungsi penulisan seperti *printChar* dan *printInt* memungkinkan data ditulis ke file secara terformat. Dengan pendekatan ini, mesin karakter memberikan kemampuan abstraksi tinggi untuk pengelolaan aliran data karakter, menyederhanakan operasi baca-tulis di berbagai bagian program. Hal ini memastikan fleksibilitas dan modularitas dalam pengembangan aplikasi yang memerlukan interaksi dengan file eksternal.

3.4 Struktur Data Mesin Kata

Pada pengerjaan program PurrMart, kami menggunakan struktur data **Mesin Kata** untuk membaca dan mengelola kata-kata dari pita atau masukan. Mesin kata ini menggunakan variabel *currentWord* untuk menyimpan kata yang sedang diproses, serta variabel *EndWord* untuk menandai akhir pembacaan kata. Kata yang diolah didefinisikan dalam tipe data *Word*, yang memiliki atribut *TabWord* berupa array karakter dan *Length* untuk menyimpan panjang kata.

Struktur data **Mesin Kata** berfungsi untuk mengelola proses pembacaan kata-kata dari masukan standar (*stdin*) atau dari file eksternal. Proses dimulai dengan fungsi *STARTWORD* atau *STARTWORDFILE*, yang mempersiapkan mesin untuk membaca kata pertama dari pita atau file. Untuk membaca kata berikutnya, fungsi *ADVWORD* digunakan untuk memajukan pita hingga kata selanjutnya ditemukan. Kata yang sudah terbaca akan disimpan dalam variabel *CurrentWord* dan dapat diakses atau dimanipulasi sesuai kebutuhan.

Mesin kata memainkan peran penting ketika aplikasi membutuhkan pengolahan input berupa kata-kata atau kalimat. Contohnya adalah pembacaan kata dari file menggunakan *STARTWORDFILE*, yang membuka file tertentu di folder *saves* dan membaca kata pertama dari file tersebut. Jika pembacaan di folder *saves* dan membaca kata pertama dari file tersebut. Jika pembacaan dilanjutkan, fungsi *ADVWORDFILE* akan membaca kata berikutnya hingga akhir file tercapai, ditandai dengan variabel *EndWord* bernilai true. Mesin ini juga mendukung pembacaan kalimat lengkap melalui fungsi *STARTLINE*, yang mengakuisisi baris masukan termasuk spasi sebagian bagian dari kata.

File *mesinkata.h* dan *mesinkata.c* membentuk ADT **Mesin Kata** yang berisi definisi struktur data *Word* serta implementasi fungsi-fungsi pendukung seperti *CopyWord*, *wordToString*, dan *wordtoint*. Fungsi *wordToString* mengonversi kata yang tersimpan di *currentWord* menjadi string C standar, sedangkan *wordtoint* digunakan untuk mengubah kata menjadi bilangan bulat. Selain itu, fungsi-fungsi seperti *IgnoreBlanks* membantu mengabaikan karakter spasi saat membaca kata.

Dengan pendekatan ini, struktur data **Mesin Kata** memberikan fleksibilitas tinggi dalam pengolahan input berbasis kata. Mesin ini dirancang untuk dapat menangani berbagai skenario, mulai dari pembacaan input pengguna hingga pengolahan file eksternal. Modularitas dan abstraksi yang ditawarkan oleh mesin kata memastikan penggunaannya dapat dengan mudah diterapkan di berbagai bagian aplikasi tanpa perlu mengetahui detail implementasi. Hal ini menjadikan **Mesin Kata** sebagai komponen penting dalam pengelolaan input dan data teks dalam program.

3.5 Struktur Data Queue

Pada pengerjaan program PurrMart, kami menggunakan struktur data *Queue* (*struct Queue*). Struktur data ini merepresentasikan antrian berbasis buffer melingkar yang digunakan untuk menyimpan data secara terorganisir dengan aturan *First In First Out*. Elemen yang disimpan dalam buffer memiliki tipe *char**, yaitu string dengan panjang maksimal yang didefinisikan oleh konstanta *QUEUE_ITEM_MAX_LEN*. Struktur ini digunakan untuk mengelola data antrian dalam aplikasi secara efisien.

Struktur data **Queue** menyimpan informasi dalam beberapa atribut utama. Atribut pertama adalah *buffer*, yaitu array status yang berfungsi sebagai tempat penyimpanan elemen-elemen antrian. Selanjutnya, atribut *IDX_HEAD* digunakan untuk menunjuk elemen pertama (head) dari antrian, sementara *IDX_TAIL* menunjuk elemen terakhir (tail) dari antrian. Kedua atribut ini bekerja secara melingkar untuk memastikan penggunaan ruang penyimpanan tetap optimal. Selain itu, atribut *QUEUE_MAX_CAPACITY* menentukan kapasitas maksimum elemen yang dapat disimpan dalam antrian. Dengan struktur ini, elemen dalam antrian dapat dimanipulasi secara efisien sesuai kebutuhan aplikasi.

Queue mulai berperan sejak awal program ketika antrian diinisialisasi menggunakan fungsi *CreateQueue*. Fungsi ini mengalokasikan memori untuk setiap elemen dalam buffer dan mengatur nilai *IDX_HEAD* serta *IDX_TAIL* menjadi *IDX_UNDEF*, menandakan bahwa antrian dalam kondisi kosong. Ketika elemen baru ditambahkan, fungsi *enqueue* digunakan untuk menambahkan elemen pada posisi *TAIL*. Jika antrian kosong, indeks *HEAD* dan *TAIL* akan diatur ke indeks pertama, jika tidak, *IDX_TAIL* diperbarui menggunakan operasi melingkar. Sebaliknya, fungsi *dequeue* digunakan untuk menghapus elemen pada posisi *HEAD* dengan aturan *First In First Out*. Jika antrian menjadi kosong setelah penghapusan, indeks *HEAD* dan *TAIL* akan diatur kembali ke *IDX_UNDEF*.

File *queue.h* dan *queue.c* membentuk sebuah ADT **Queue** yang berisi definisi struktur data **Queue** serta implementasi fungsi-fungsi pendukung. Fungsi seperti *isEmpty* dan *isFull* digunakan untuk memeriksa status antrian, sedangkan fungsi *length* menghitung jumlah elemen efektif yang sedang tersimpan. Untuk memantau isi antrian, fungsi *displayQueue* menampilkan elemen-elemen antrian dalam format terstruktur. Dengan fungsi-fungsi ini, pengelolaan data antrian menjadi lebih terorganisir dan fleksibel.

Dengan pendekatan ini, struktur data **Queue** memberikan solusi yang efisien untuk menyimpan dan mengelola data dalam urutan *First In First Out*. Implementasi yang modular dan abstrak memungkinkan penggunaannya di berbagai bagian program tanpa mengungkap detail internal. Hal ini menjadikan **Queue** sebagai komponen penting dalam aplikasi yang membutuhkan pengelolaan data antrian secara terstruktur dan efisien.

3.6 Struktur Data Stack

Pada pengerjaan program PurrMart, kami menggunakan struktur data **Stack** (*struct Stack*) untuk mengelola data dalam urutan *LIFO* (*Last In, First Out*). Struktur data ini memanfaatkan **array statis** untuk menyimpan elemen-elemen stack, dengan indeks *Top* yang menunjukkan posisi elemen terakhir atau puncak stack. Elemen yang disimpan memiliki tipe data *infostack*, yang dapat disesuaikan dengan kebutuhan program.

Struktur data **Stack** menyimpan informasi melalui beberapa atribut utama. Atribut pertama adalah *array T*, yang digunakan sebagai penampung elemen-elemen stack. Atribut kedua adalah *Top*, yaitu indeks yang menunjukkan elemen paling atas dalam stack. Saat stack kosong, *Top* akan bernilai *NilStack*, menandakan bahwa tidak ada elemen yang tersimpan. Selain itu, kapasitas maksimum stack ditentukan oleh konstanta *MaxStackEl*, yang memastikan batas maksimal elemen yang dapat disimpan di dalam stack. Dengan struktur ini, data pada stack dapat ditambahkan dan dihapus secara terstruktur menggunakan aturan *Last In, First Out*.

Stack mulai berperan sejak awal program ketika diinisialisasi menggunakan fungsi *CreateEmptyStack*. Fungsi ini mengatur nilai *Top* menjadi *NilStack*, menandakan bahwa stack dalam keadaan kosong. Elemen baru dapat ditambahkan ke stack menggunakan fungsi *Push*, yang menambah elemen pada posisi puncak dan memperbarui nilai *Top*. Sebaliknya, elemen di puncak stack dapat dihapus menggunakan fungsi *Pop*, yang mengembalikan nilai elemen tersebut dan mengurangi nilai *Top*. Selain itu, fungsi *IsEmptyStack* dan *IsFull* digunakan untuk memeriksa apakah stack dalam keadaan kosong atau penuh.

File *stack.h* dan *stack.c* membentuk sebuah ADT **Stack** yang berisi definisi struktur data dan implementasi fungsi-fungsi pendukung. Fungsi seperti *LengthStack* menghitung jumlah elemen efektif di dalam stack, sedangkan fungsi *ReverseStack* memungkinkan pembalikan

urutan elemen pada stack dengan memanfaatkan stack tambahan sebagai penyimpanan sementara. Fungsi-fungsi ini memastikan bahwa pengelolaan data pada stack dapat dilakukan dengan cara yang efisien dan terstruktur

Implementasi yang modular dan abstrak memungkinkan penggunaan ulang fungsi-fungsi stack di berbagai bagian program. Hal ini menjadikan Stack sebagai komponen penting dalam aplikasi yang membutuhkan pengelolaan data secara terorganisir dan fleksibel.⁶

4 Program Utama

Pada program aplikasi PurrMart, alur *program* dijalankan dan dimulai dari 'main.c'. Program ini akan menuju ke *welcome menu* dan memerintahkan pengguna untuk input beberapa *COMMAND*, yaitu *START*, *LOAD*, *HELP*, dan *EXIT* sebagai masukan pertama dari aplikasi PurrMart. *COMMAND START* berfungsi untuk membuka aplikasi dengan *file* konfigurasi default yang sudah berisi daftar jumlah barang, harga barang, nama barang, banyak user, uang user, username user, dan password user. Sedangkan *COMMAND LOAD* berfungsi untuk memuat *save file* yang berisi daftar jumlah barang, harga barang, nama barang, banyak user, uang user, username user, dan password user. Sedangkan, *COMMAND HELP* berfungsi untuk menampilkan daftar command yang dapat dieksekusi beserta deskripsinya. Apabila file konfigurasi tidak berhasil dimuat, pengguna akan kembali ke *welcome menu* dan proses yang sama akan berulang. Apabila file konfigurasi telah berhasil dibaca, program akan menampilkan login menu dalam aplikasi yang berisi *COMMAND REGISTER*, *COMMAND LOGIN*, dan *COMMAND EXIT*. *COMMAND REGISTER* berfungsi untuk menambahkan sebuah List User dengan meminta input username dan password yang diinput pengguna. *COMMAND LOGIN* berfungsi agar pengguna dapat masuk ke dalam *command* utama aplikasi dengan menggunakan akun yang telah di-*register* atau dengan data akun (username-password) yang ada dalam file konfigurasi yang dimuat. Setelah pengguna berhasil login, pengguna akan tertuju ke dalam program utama dalam aplikasi yang berisi command-command utama dalam aplikasi.

Program ini memiliki beberapa **command** utama setelah **file konfigurasi telah berhasil dibaca**.

1. **WORK** : Menampilkan daftar pekerjaan dan rincian pendapatan pekerjaan dan durasi pekerjaan.
2. **WORK CHALLENGE** : Memberi pilihan *challenge* untuk memperoleh uang
3. **STORE LIST** : Menampilkan daftar barang yang tersedia di toko.
4. **STORE REQUEST** : Memasukkan nama barang yang ingin dimasukkan ke dalam antrian
5. **STORE SUPPLY** : Menentukan barang di antrian apakah dimasukkan ke dalam toko, sesuai dengan pilihan pengguna.
6. **STORE REMOVE** : Melakukan penghapusan barang yang ada di toko.
7. **LOGOUT** : Keluar dari akun
8. **SAVE** : Menyimpan status aplikasi saat ini ke dalam file yang dituju.
9. **EXIT** : Keluar dari aplikasi **dengan opsi untuk menyimpan data sesi saat keluar**.
10. **HELP** : Menampilkan daftar *command* yang dapat digunakan beserta fungsinya
11. **BIOWEAPON** : Membuat sebuah senjata biologis dengan masukan sekuens DNA dan kode rahasia yang sesuai
12. **BACK** : Kembali dari *main menu* ke *login menu*

5 Algoritma-Algoritma Menarik

5.1 Konversi DNA ke RNA

Dalam program `bioweapon.c`, implementasi program tersebut menggunakan fungsi untuk melakukan konversi dari DNA ke RNA, yaitu :

```
char* DNAtoRNA(const char* dna);
```

Terdapat lagi sebuah fungsi konversi dalam program tersebut, yaitu fungsi konversi RNA ke protein, yaitu :

```
char RNAtoProtein(const char* codon);
```

Hal yang menjadi keunikan dalam fungsi konversi DNA ke RNA adalah penggunaan pointer `char*` untuk mengembalikan string RNA yang dihasilkan, berbeda dengan fungsi konversi RNA ke protein yang hanya menggunakan `char` saja. Walaupun sama-sama berfungsi untuk melakukan konversi, tetapi terdapat perbedaan dalam penggunaan `char` dan `char*`. Penggunaan `char*` dalam fungsi konversi DNA ke RNA dibutuhkan karena fungsi tersebut menghasilkan string yang berupa array karakter dengan panjang yang sama dengan inputnya. Karena panjang dari input tidak diketahui sebelumnya, pointer memungkinkan untuk menangani string dinamis dengan panjang yang belum diketahui. Tidak menggunakan `char` karena tipe `char` hanya dapat menyimpan satu karakter, sehingga tidak cukup mewakili seluruh urutan RNA.

6 Data Test

6.1 Data Test START

Pada saat program `PurrMart` pertama kali dimulai, akan ditampilkan pesan pembuka beserta *command-command* yang dapat dipanggil, salah satunya adalah *command START*. pengguna dapat memasukkan *command START* atau angka 1 untuk mengkonfigurasi program `PurrMart` dengan file konfigurasi default. Ketika *command* dijalankan, program akan membaca file *default_config.txt* yang berisi konfigurasi default.

Pada program, terdapat state yang menandakan sesi dari program `PurrMart`. Apabila ingin memasuki sesi, pengguna perlu melakukan konfigurasi dengan *START* atau *LOAD*. Setelah *command* berhasil dijalankan state sesi akan berubah sehingga program masuk ke sesi yang dapat memasukkan *command-command* baru untuk memulai aplikasi *PurrMart*. *Command START* hanya dapat diakses ketika belum memasuki sesi. Apabila sudah memasuki sesi, maka pengguna harus memberikan *command* lain dan program akan mengeluarkan pemberitahuan bahwa *command* tidak bisa dieksekusi.

Gambar 6.1.1 Tampilan awal ketika program dijalankan

STEI- ITB	IF2111 TB 01 03	Halaman 12 dari 45 halaman
Template dokumen ini dan informasi yang dimilikinya adalah milik Sekolah Teknik Elektro dan Informatika ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Sekolah Teknik Elektro dan Informatika ITB.		

Gambar 6.2.1 Tampilan ketika load dipanggil dengan nama file yang valid

```
MASUKKAN COMMAND: LOAD
Masukkan nama file yang ingin anda load : tubes.txt

===== LOAD MENU =====
File tubes.txt berhasil dimuat.

=====
                        LOGIN MENU
=====
1. REGISTER
2. LOGIN
3. LOGOUT
4. EXIT
5. HELP
=====
```

Gambar 6.2.2 Tampilan ketika load dipanggil dengan nama file yang tidak valid

```
=====
                        WELCOME MENU
=====
1. START
2. LOAD
3. EXIT
4. HELP
=====

MASUKKAN COMMAND: LOAD
Masukkan nama file yang ingin anda load : fileasal.txt

===== LOAD MENU =====
File fileasal.txt tidak ditemukan.

Kembali ke Welcome Menu.
```

6.3 Data Test HELP

Pengguna aplikasi dapat memanggil *command* HELP atau angka 4 (*welcome menu* dan *login menu*) atau angka 10 (*main menu*) untuk menunjukkan daftar *command* yang dapat digunakan di bagian program beserta fungsinya. *Command* HELP bisa dipanggil di ketiga bagian program, yaitu *Welcome Menu*, *Login Menu*, dan *Main Menu*. Setiap bagian program akan mengirimkan *command* sesuai dengan daftar *command* yang dapat digunakan disaat itu.

a. Welcome Menu

Gambar 6.3.1 Tampilan HELP pada Welcome Menu

```
=====
                        WELCOME MENU
=====
1. START
2. LOAD
3. EXIT
4. HELP
=====

MASUKKAN COMMAND: 4

=====[ Welcome Help Menu PURRMART ]=====
START -> Untuk masuk sesi baru
LOAD -> Untuk memulai sesi berdasarkan file konfigurasi
EXIT -> Untuk keluar dari program
```

b. Login Menu

Gambar 6.3.2 Tampilan HELP pada Login Menu

```
=====
                        LOGIN MENU
=====
1. REGISTER
2. LOGIN
3. LOGOUT
4. EXIT
5. HELP
=====

MASUKKAN COMMAND: HELP

=====[ Login Help Menu PURRMART ]=====
REGISTER -> Untuk melakukan pendaftaran akun baru
LOGIN -> Untuk masuk ke dalam akun dan memulai sesi
LOGOUT -> Untuk keluar dari sesi
EXIT -> Untuk keluar dari program
```

c. Main Menu

Gambar 6.3.3 Tampilan HELP pada Main Menu

```
=====
MASUKKAN COMMAND:  help

=====[ Main Help Menu PURRMART ]=====
WORK -> Untuk bekerja
WORK CHALLENGE -> Untuk mengerjakan challenge
STORE LIST -> Untuk melihat barang-barang di toko
STORE REQUEST -> Untuk meminta penambahan barang
STORE SUPPLY -> Untuk menambahkan barang dari permintaan
STORE REMOVE -> Untuk menghapus barang
LOGOUT -> Untuk keluar dari sesi
SAVE -> Untuk menyimpan state ke dalam file
EXIT -> Untuk keluar dari program
BACK -> Untuk kembali ke login menu tanpa logout
BIOWEAPON -> Untuk membuat senjata biologis dengan kode rahasia
=====
```

6.4 Data Test REGISTER

Pengguna aplikasi dapat memanggil *command* REGISTER atau angka 1 untuk mendaftarkan akun dengan username yang unik (tidak boleh sama dengan username yang sudah tercatat di *file*). Register dilakukan untuk membuat sebuah akun baru, sebelum dapat masuk ke dalam aplikasi melalui LOGIN.

Gambar 6.4.1 Tampilan REGISTER yang berhasil

```
=====
                        LOGIN MENU
=====
1. REGISTER
2. LOGIN
3. LOGOUT
4. EXIT
5. HELP
=====

MASUKKAN COMMAND: REGISTER
Masukkan username baru: hugo
Masukkan password baru: h
Pendaftaran berhasil untuk username: hugo
```


Gambar 6.4.2 Tampilan REGISTER yang gagal karena username sudah ada

```
=====
                        LOGIN MENU
=====
1. REGISTER
2. LOGIN
3. LOGOUT
4. EXIT
5. HELP
=====

MASUKKAN COMMAND: REGISTER
Masukkan username baru: hugo
Masukkan password baru: h
Username sudah ada, silakan gunakan username lain.
```

6.5 Data Test LOGIN

Pengguna aplikasi dapat memanggil *command* LOGIN untuk masuk ke aplikasi dengan akun yang sudah di-register sebelumnya oleh pengguna atau dengan data akun (username-password) yang telah tercantum dalam file konfigurasi yang dijalankan.

Gambar 6.5.1 Tampilan LOGIN jika berhasil

```
=====
                        LOGIN MENU
=====
1. REGISTER
2. LOGIN
3. LOGOUT
4. EXIT
5. HELP
=====

MASUKKAN COMMAND: login
Masukkan username: ai
Masukkan password: owo
Login berhasil sebagai ai dengan uang 1000.
```

Gambar 6.5.2 Tampilan LOGIN jika gagal

```
=====
                        LOGIN MENU
=====
1. REGISTER
2. LOGIN
3. LOGOUT
4. EXIT
5. HELP
=====

MASUKKAN COMMAND: login
Masukkan username: akun1
Masukkan password: agentp
Login gagal. Username atau password salah.
```

Gambar 6.5.3 Tampilan LOGIN apabila belum ada pengguna dengan username tersebut

```
=====
                        LOGIN MENU
=====
1. REGISTER
2. LOGIN
3. LOGOUT
4. EXIT
5. HELP
=====

MASUKKAN COMMAND: login
Masukkan username: akun1
Masukkan password: agentp
Login gagal. Username atau password salah.
```

Gambar 6.5.4 Tampilan LOGIN apabila belum LOGOUT

```
<<< BACK
=====

MASUKKAN COMMAND: <<<

=====
                        LOGIN MENU
=====
1. REGISTER
2. LOGIN
3. LOGOUT
4. EXIT
5. HELP
=====

MASUKKAN COMMAND: 2
Anda masih tercatat sebagai AgentF. Silakan LOGOUT terlebih dahulu.

=====
                        LOGIN MENU
=====
1. REGISTER
2. LOGIN
3. LOGOUT
4. EXIT
5. HELP
=====

MASUKKAN COMMAND: 
```

6.6 Data Test WORK

Pengguna aplikasi dapat memanggil *command* WORK sebagai alternatif untuk pengguna mendapatkan uang dengan melakukan pekerjaan yang dipilih. Setiap pekerjaan memiliki waktu tunggu yang berbeda-beda dan dengan nominal pendapatan yang berbeda-beda. Dan selama pengguna sedang bekerja, maka sistem tidak bisa digunakan hingga pekerjaan selesai dilakukan.

Gambar 6.6.1 Tampilan WORK saat pengguna bekerja

```
MASUKKAN COMMAND: work

=== DAFTAR PEKERJAAN ===
1. Evil Lab Assistant (pendapatan=1000, durasi=2s)
2. OWCA Hiring Manager (pendapatan=4200, durasi=3s)
3. Cikapundunginator Caretaker (pendapatan=7000, durasi=5s)
4. Mewing Specialist (pendapatan=10000, durasi=10s)
5. Inator Connoisseur (pendapatan=9997, durasi=8s)
=====
Masukkan nomor pekerjaan yang dipilih: 1
..
Pekerjaan selesai, +1000 rupiah telah ditambahkan ke akun Anda.
```

Gambar 6.6.2 Tampilan WORK menambahkan uang pengguna

```
MASUKKAN COMMAND: 1

=== DAFTAR PEKERJAAN ===
1. Evil Lab Assistant (pendapatan=1000, durasi=2s)
2. OWCA Hiring Manager (pendapatan=4200, durasi=3s)
3. Cikapundunginator Caretaker (pendapatan=7000, durasi=5s)
4. Mewing Specialist (pendapatan=10000, durasi=10s)
5. Inator Connoisseur (pendapatan=9997, durasi=8s)
=====
Masukkan nomor pekerjaan yang dipilih: 5
.....
Pekerjaan selesai, +9997 rupiah telah ditambahkan ke akun Anda.
```

Gambar 6.7.1 Tampilan WORK CHALLENGE pada Main Menu

```
C > Tugas-Besar-IF2111-K1-03-main > saves > default_config.txt
1 7
2 250 Invisibility Cloak
3 300 Arc Reactor
4 450 Eye of Agamotto
5 400 Elder Wand
6 450 Stormbreaker
7 500 Infinity Gauntlet
8 300 Vibranium Shield
9 5
10 100000 AgentF tes123
11 5000 User2 tes123
12 1000 User3 tes123
13 1500 user4 tes123
14 2000 Aqmar tes123
15
```

Gambar 6.7.2 Tampilan jika uang tidak cukup untuk memainkan challenge

```
C > Tugas-Besar-IF2111-K1-03-main > saves > saveafterwork.txt
1 7
2 250 Invisibility Cloak
3 300 Arc Reactor
4 450 Eye of Agamotto
5 400 Elder Wand
6 450 Stormbreaker
7 500 Infinity Gauntlet
8 300 Vibranium Shield
9 5
10 109997 AgentF tes123
11 5000 User2 tes123
12 1000 User3 tes123
13 1500 user4 tes123
14 2000 Aqmar tes123
15
```

6.7 Data Test WORK CHALLENGE

Pengguna program dapat memanggil command WORK CHALLENGE atau angka 2 untuk melakukan challenge-challenge yang ada di OWCA. Pemain dengan jumlah uang tertentu dapat memainkan challenge.

Gambar 6.7.1 Tampilan list WORK CHALLENGE

```
Selamat datang di PURRMART!

=====
MAIN MENU
=====
1. WORK
2. WORK CHALLENGE
3. STORE LIST
4. STORE REQUEST
5. STORE SUPPLY
6. STORE REMOVE
7. LOGOUT
8. SAVE
9. EXIT
10. HELP
11. BIOWEAPON
<<< BACK
=====

MASUKKAN COMMAND: 2

=====
WORK CHALLENGE LIST
=====
1. TEBAK ANGKA (PLAYING COST = 200)
2. WORDL3 (PLAYING COST = 500)
3. QUANTUM WORDL3 (PLAYING COST = 750)
=====

APA CHALLENGE YANG HENDAK KAMU MAINKAN: |
```

Gambar 6.7.1 Tampilan WORK CHALLENGE apabila saldo tidak cukup

```
MASUKKAN COMMAND: WORK CHALLENGE

=====
                      WORK CHALLENGE LIST
=====
1. TEBAK ANGKA (PLAYING COST = 200)
2. W0RDL3 (PLAYING COST = 500)
3. QUANTUM W0RDL3 (PLAYING COST = 750)
=====

APA CHALLENGE YANG HENDAK KAMU MAINKAN: W0RDL3
Saldo Anda tidak cukup untuk bermain W0RDL3.
```

6.7.1 Data Test Tebak Angka

Pengguna yang telah memanggil command WORK CHALLENGE dan telah memilih challenge Tebak Angka akan menghadapi permainan menebak angka yang ditentukan program.

Gambar 6.7.1.1 Tampilan apabila memilih Tebak Angka

```
=====
                      WORK CHALLENGE LIST
=====
1. TEBAK ANGKA (PLAYING COST = 200)
2. W0RDL3 (PLAYING COST = 500)
3. QUANTUM W0RDL3 (PLAYING COST = 750)
=====

APA CHALLENGE YANG HENDAK KAMU MAINKAN: 1
SELAMAT DATANG DI TEBAK ANGKA!

BIAYA BERMAIN SEBESAR 200 TELAH DIKURANGI DARI SALDO ANDA! SEMOGA ANDA MENANG!

Uji keberuntungan Anda dengan menebak angka!
Cheat: 30
Tebakan angka: █
```

Gambar 6.7.1.2. Tampilan apabila pengguna langsung menang Tebak Angka

C > Tugas-Besar-IF2111-K1-03-main > saves > default_config.txt
1 7
2 250 Invisibility Cloak
3 300 Arc Reactor
4 450 Eye of Agamotto
5 400 Elder Wand
6 450 Stormbreaker
7 500 Infinity Gauntlet
8 300 Vibranium Shield
9 5
10 100000 AgentF tes123
11 5000 User2 tes123
12 1000 User3 tes123
13 1500 user4 tes123
14 2000 Aqmar tes123
15

C > Tugas-Besar-IF2111-K1-03-main > saves > saveaftertebak.txt
1 7
2 250 Invisibility Cloak
3 300 Arc Reactor
4 450 Eye of Agamotto
5 400 Elder Wand
6 450 Stormbreaker
7 500 Infinity Gauntlet
8 300 Vibranium Shield
9 5
10 100300 AgentF tes123
11 5000 User2 tes123
12 1000 User3 tes123
13 1500 user4 tes123
14 2000 Aqmar tes123
15

```

APA CHALLENGE YANG HENDAK KAMU MAINKAN: 1
SELAMAT DATANG DI TEBAK ANGKA!

BIAYA BERMAIN SEBESAR 200 TELAH DIKURANGI DARI SALDO ANDA! SEMOGA ANDA MENANG!

Uji keberuntungan Anda dengan menebak angka!
Cheat: 30
Tebakan angka: 30
Tebakanmu benar! +500 rupiah telah ditambahkan ke akun anda.

```

Catatan : Lihat data AgentF yang mendapat tambahan 300 coins (-200 cost masuk dan +500 akibat menang)

Gambar 6.7.1.3. Tampilan apabila pengguna gagal memenangkan challenge Tebak Angka

```
Uji keberuntungan Anda dengan menebak angka!  
Cheat: 14  
Tebakan angka: 1  
Lebih besar  
Tebakan: 1  
Lebih besar  
Tebakan: 1  
Lebih besar  
Tebakan: 1  
Lebih besar  
Tebakan: 1  
Lebih besar  
Tebakan: 1  
Lebih besar  
Tebakan: 1  
Lebih besar  
Tebakan: 1  
Lebih besar  
Tebakan: 1  
Lebih besar  
Tebakan: 1  
Anda tidak beruntung wkwkw. +0 rupiah telah ditambahkan ke akun anda.
```

6.7.2 Data Test W0RDL3

Pengguna yang telah memanggil command WORK CHALLENGE dan telah memilih challenge W0RDL3 akan menghadapi permainan menebak kata sepanjang lima karakter.

Gambar 6.7.2.1 Tampilan apabila memilih challenge W0RDL3

```
=====
                        WORK CHALLENGE LIST
=====
1. TEBAK ANGKA (PLAYING COST = 200)
2. W0RDL3 (PLAYING COST = 500)
3. QUANTUM W0RDL3 (PLAYING COST = 750)
=====

APA CHALLENGE YANG HENDAK KAMU MAINKAN: W0RDL3
WELCOME TO W0RDL3!

500 COINS HAVE BEEN SPENT TO JOIN THIS CHALLENGE. GO FOR THE WIN!

YOU HAVE 5 CHANCES TO ANSWER BEFORE YOU LOSE!

- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -

Masukan kata tebakn Anda: |
```

Gambar 6.7.2.2. Tampilan apabila pengguna langsung memenangkan WORDL3

```

Masukan kata tebakan Anda: aaaaa
Hasil:
A% A A% A% A%
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -

Masukan kata tebakan Anda: LUCKY
Hasil:
A% A A% A% A%
L% U% C* K% Y%
- - - - -
- - - - -
- - - - -
- - - - -

Masukan kata tebakan Anda: DANCE
Hasil:
A% A A% A% A%
L% U% C* K% Y%
D A N C E
- - - - -
- - - - -
- - - - -

Selamat, Anda menang! Anda mendapatkan 3000 coin!

```

Gambar 6.7.2.3. Tampilan apabila pengguna gagal memenangkan challenge WORDL3

```

Masukan kata tebakan Anda: AAAAA
Hasil:
A% A% A% A% A%
A% A% A% A% A%
A% A% A% A% A%
A% A% A% A% A%
A% A% A% A% A%
A% A% A% A% A%

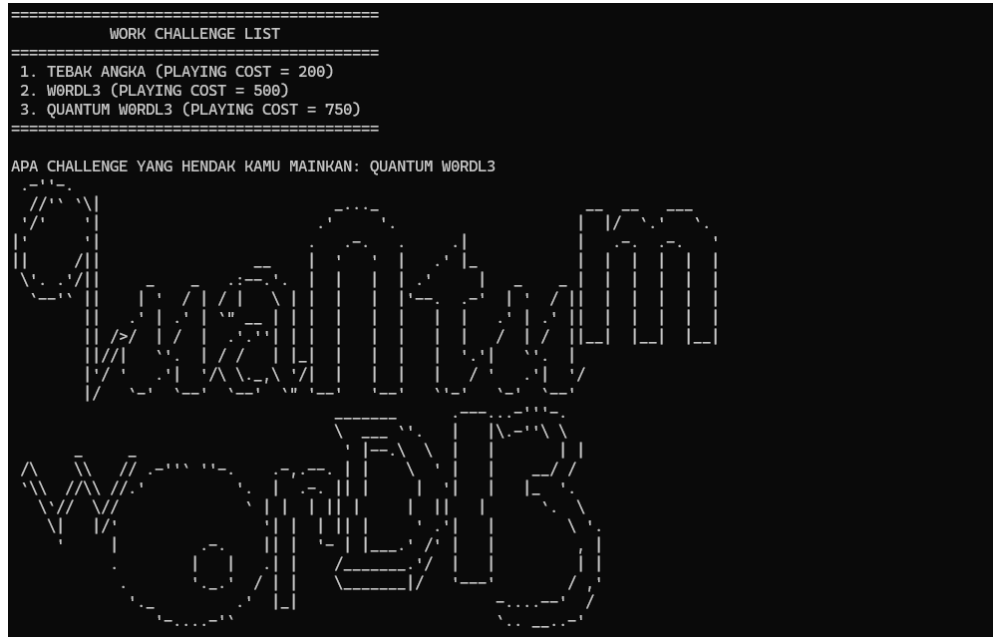
Boo! Anda kalah.

```


6.7.3 Data Test Quantum WORDL3

Pengguna yang telah memanggil command WORK CHALLENGE dan telah memilih challenge Quantum WORDL3 akan menghadapi permainan menebak kata sepanjang lima karakter. Permainan ini memungkinkan pemain menebak empat kata sekaligus dalam satu waktu.

Gambar 6.7.2.1 Tampilan apabila memilih Quantum WORDL3



Gambar 6.7.2.2. Tampilan apabila pengguna memenangkan challenge Quantum WORDL3

```
Results for 'TRULY':

Try 1: L% U% C* K% Y%  L* U* C% K% Y  L% U% C* K% Y%  L* U% C* K* Y%
Try 2: B% E* A* C H%  B% E% A% C% H%  B% E% A% C* H%  B% E% A% C H%
Try 3: C* R% O% W% N*  C% R O% W% N%  C R O W N  C* R% O% W% N%
Try 4: S% L% I% C K%  S% L* I% C% K%  C R O W N S L I C K
Try 5: D A N C E  D% A% N% C% E%  C R O W N S L I C K
Try 6: D A N C E  B% U* I% L D%  C R O W N S L I C K
Try 7: D A N C E  L* E% A% K% Y  C R O W N S L I C K
Try 8: D A N C E  T R U L Y  C R O W N S L I C K
Try 9: - - - - -  - - - - -  - - - - -  - - - - -

Word 1: SOLVED!
Word 2: SOLVED!
Word 3: SOLVED!
Word 4: SOLVED!

Game Over!
Words solved: 4/4
Attempts used: 8/9
Congratulations! You solved all words! You've earned 15,000 coins as your reward. Enjoy y
our victory!
```

Gambar 6.7.2.3. Tampilan apabila pengguna gagal memenangkan challenge Tebak Angka

```
Results for 'SUSAH':

Try 1: A% A% A A% A%  A% A% A% A% A%  A% A% A% A% A%  A% A% A% A% A%
Try 2: A% A% A A% A%  A% A% A% A% A%  A% A% A% A% A%  A% A% A% A% A%
Try 3: A% A% A A% A%  A% A% A% A% A%  A% A% A% A% A%  A% A% A% A% A%
Try 4: A% A% A A% A%  A% A% A% A% A%  A% A% A% A% A%  A% A% A% A% A%
Try 5: A% A% A A% A%  A% A% A% A% A%  A% A% A% A% A%  A% A% A% A% A%
Try 6: A% A% A A% A%  A% A% A% A% A%  A% A% A% A% A%  A% A% A% A% A%
Try 7: A% A% A A% A%  A% A% A% A% A%  A% A% A% A% A%  A% A% A% A% A%
Try 8: A% A% A A% A%  A% A% A% A% A%  A% A% A% A% A%  A% A% A% A% A%
Try 9: S% U% S% A* H%  S% U S% A% H%  S% U S% A% H%  S% U% S% A% H%

Word 1: Not solved yet
Word 2: Not solved yet
Word 3: Not solved yet
Word 4: Not solved yet

Game Over!
Words solved: 0/4
Attempts used: 9/9
Better luck next time!
The target words were: LEAKY BUILD LUCKY CROWN
```

6.8 Data Test BIOWEAPON

Pengguna program ini dapat memanggil command BIOWEAPON (11) untuk membuat senjata biologis dengan sekuens DNA dan kode rahasia yang sesuai.

Gambar 6.8.1 Tampilan apabila kode rahasia sesuai

```
11. BIOWEAPON
<<< BACK
=====

MASUKKAN COMMAND: 11
Masukan nama senjata biologis: Batuksius fififafae
Masukan sekuens DNA: TAAATGATGAGATAACCATAACCGGGCCGCAATT
Masukan kode rahasia: SIGMA
TAA ATG ATG AGA TAA CCA TAC CGG GCC GCA ATT
AUU UAC UAC UCU AUU GGU AUG GCC CGG CGU UAA
IYYSIGMARR
Senjata biologis 'Batuksius' mengandung kode, barang akan ditambahkan ke dalam queue!
```

Gambar 6.8.2 Tampilan apabila kode rahasia tidak sesuai

```
MASUKKAN COMMAND: 11
Masukan nama senjata biologis: HEHEHEHE
Masukan sekuens DNA: ATGCGCATGCT
Masukan kode rahasia: SIGMA
ATG CGC ATG CT
UAC GCG UAC GA
YAY
TRT
RVR
Kode rahasia tidak ditemukan, maka senjata biologis sudah disabotase, barang ditolak!
```

6.9 Data Test *STORE LIST*

Pengguna program ini dapat memanggil command *STORE LIST* untuk menampilkan list barang yang tersedia di toko.

*Gambar 6.9.1 Tampilan *STORE LIST* jika terdapat barang di toko*

```
MASUKKAN COMMAND: store list

=====
                DAFTAR BARANG DI TOKO
=====

1. Invisibility Cloak
2. Arc Reactor
3. Eye of Agamotto
4. Elder Wand
5. Stormbreaker
6. Infinity Gauntlet
7. Vibranium Shield
=====
```

Gambar 6.9.2 Tampilan STORE LIST jika tidak terdapat barang di toko

```
=====
MASUKKAN COMMAND:  store list
=====
                    TOKO SAAT INI KOSONG
=====
```

6.10 Data Test STORE REQUEST

Pengguna program ini dapat memanggil command STORE REQUEST untuk melakukan permintaan memasukkan barang baru ke dalam antrian sebelum dimasukkan ke toko. Program hanya akan menerima permintaan apabila barang yang di-request tidak ada di dalam antrian ataupun di dalam toko.

Gambar 6.10.1 Tampilan STORE REQUEST jika barang belum terdapat di antrean

```
=====
MASUKKAN COMMAND:  store request
Nama barang yang diminta: AK47
Barang AK47 dah dimasukin di antrian coyy!
=====
```

Gambar 6.10.2 Tampilan STORE REQUEST jika barang sudah terdapat di antrean

```
=====
MASUKKAN COMMAND:  4
Nama barang yang diminta: AK47
Barang dengan nama yang sama sudah ada di antrian!
=====
```

Gambar 6.10.3 Tampilan STORE REQUEST jika barang sudah terdapat di toko

```
=====
MASUKKAN COMMAND:  store request
Nama barang yang diminta: Arc Reactor
Barang dengan nama yang sama sudah ada di toko!
=====
```

6.11 Data Test STORE SUPPLY

Pengguna program ini dapat memanggil command STORE SUPPLY untuk menentukan apakah barang yang berada di antrian terdepan diterima, ditunda, atau ditolak untuk masuk ke dalam toko.

Gambar 6.11.1 Tampilan STORE SUPPLY saat tidak ada barang di antrian

```
=====
MASUKKAN COMMAND: store supply
Tidak ada barang di antrian untuk diproses.
```

Gambar 6.11.2 Tampilan STORE SUPPLY barang diterima

```
=====
MASUKKAN COMMAND: store supply
Apakah kamu ingin menambahkan barang AK47? (Terima/Tunda/Tolak) : Terima
Harga barang: 12000
AK47 dengan harga 12000 telah ditambahkan ke toko.
```

Gambar 6.11.3 Tampilan STORE SUPPLY saat barang ditunda

```
=====
MASUKKAN COMMAND: store supply
Apakah kamu ingin menambahkan barang AK47? (Terima/Tunda/Tolak) : Tunda
AK47 dikembalikan ke antrian.
```

Gambar 6.11.4 Tampilan STORE SUPPLY saat barang ditolak

```
=====
MASUKKAN COMMAND: store supply
Apakah kamu ingin menambahkan barang AK47? (Terima/Tunda/Tolak) : Tolak
AK47 dihapuskan dari antrian.
```

6.12 Data Test STORE REMOVE

Pengguna program dapat memanggil *command* STORE REMOVE untuk menghapus barang yang berada di toko.

Gambar 6.12.1 Tampilan STORE REMOVE jika barang berhasil dihapus

```
6. STORE REMOVE
7. LOGOUT
8. SAVE
9. EXIT
10. HELP
=====

MASUKKAN COMMAND: store remove
Nama barang yang akan dihapus: Arc Reactor
Arc Reactor telah berhasil dihapus.

=====
```

Gambar 6.12.2 Tampilan STORE REMOVE jika barang yang dihapus tidak ada di toko

```
MASUKKAN COMMAND: store remove
Nama barang yang akan dihapus: ak47
Toko tidak menjual ak47
```

6.13 Data Test LOGOUT

Pengguna program ini dapat memanggil *command* LOGOUT untuk user keluar dari akun yang sedang digunakan, tetapi pengguna tidak keluar dari aplikasi atau program.

Gambar 6.13.1 Tampilan LOGOUT

```
=====

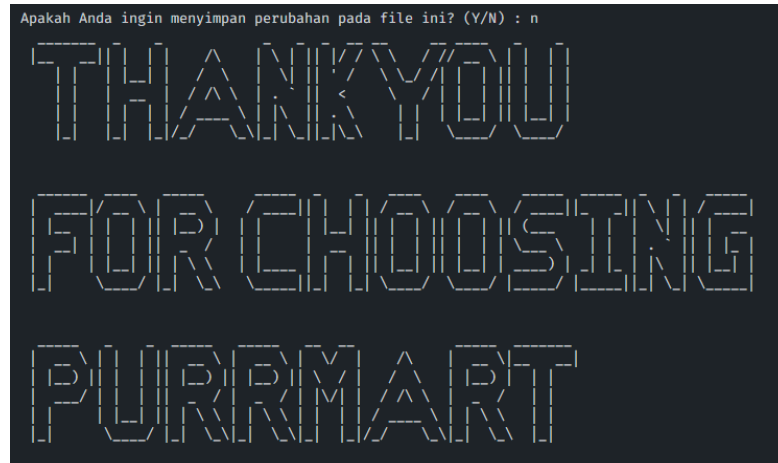
MASUKKAN COMMAND: LOGOUT
Anda telah logout.

MASUKKAN COMMAND: 
```

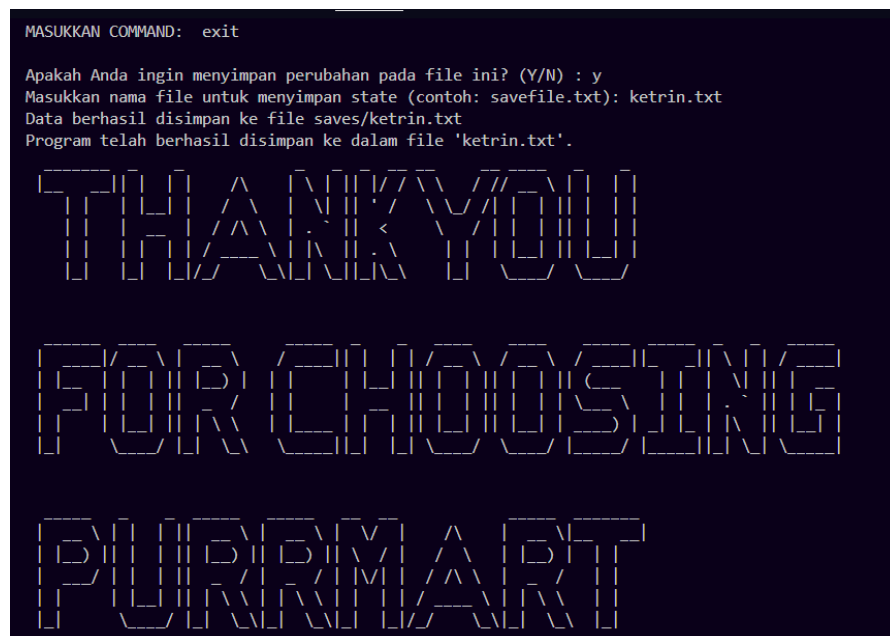
6.14 Data Test EXIT

Pengguna aplikasi dapat memanggil *command* EXIT untuk keluar dari sesi aplikasi. Namun, sebelum keluar dari aplikasi, program akan menanyakan apakah pengguna ingin melakukan SAVE terhadap aplikasi yang telah dijalankan. Jika pengguna program memilih melakukan SAVE, maka perubahan pada aplikasi akan disimpan pada suatu *file* yang diinginkan pengguna. Jika pengguna program memilih untuk tidak melakukan SAVE, maka perubahan pada aplikasi tidak akan disimpan.

Gambar 6.14.1 Tampilan EXIT jika pengguna tidak ingin menyimpan perubahan



Gambar 6.14.2 Tampilan EXIT jika pengguna ingin menyimpan perubahan



6.15 Data Test SAVE

Pengguna aplikasi dapat memanggil *command* SAVE untuk menyimpan *state* aplikasi terbaru. SAVE dapat dilakukan dengan menginput SAVE pada *command* atau SAVE setelah melakukan melakukan inputan EXIT pada *command* program. Pada kasus jika menggunakan *command* SAVE dan inputan nama file terdeteksi belum ada, maka program akan secara otomatis menyimpan *state* di file konfigurasi baru dengan nama file sesuai dengan inputan.

Gambar 6.15.1 Tampilan ketika SAVE dipanggil dengan nama file yang valid

```
=====
MAIN MENU
=====
1. WORK
2. WORK CHALLENGE
3. STORE LIST
4. STORE REQUEST
5. STORE SUPPLY
6. STORE REMOVE
7. LOGOUT
8. SAVE
9. EXIT
10. HELP
=====

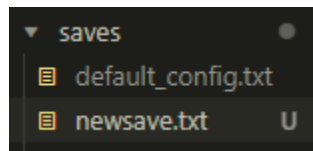
MASUKKAN COMMAND: save
Masukkan nama file untuk menyimpan state (contoh: savefile.txt): newsave.txt
Data berhasil disimpan ke file saves/newsave.txt
Program telah berhasil disimpan ke dalam file 'newsave.txt'.
```

Gambar 6.15.2 Tampilan ketika SAVE dipanggil dengan nama file yang tidak valid

```
=====
MAIN MENU
=====
1. WORK
2. WORK CHALLENGE
3. STORE LIST
4. STORE REQUEST
5. STORE SUPPLY
6. STORE REMOVE
7. LOGOUT
8. SAVE
9. EXIT
10. HELP
=====

MASUKKAN COMMAND: save
Masukkan nama file untuk menyimpan state (contoh: savefile.txt): newsave
Mohon masukkan nama file yang berakhiran dengan .txt
```

Gambar 6.15.3 Tampilan file hasil SAVE



7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	START	Memeriksa apakah file config default berhasil dibaca	Memberikan <i>command</i> START atau menginput angka 1 pada saat pertama kali memulai program PurrMart	START	Gambar 6.1.1 Gambar 6.1.2	Sesuai yang diharapkan
2	LOAD	Memeriksa apakah <i>file</i> berhasil dibaca	Memberikan <i>command</i> LOAD atau menginput angka 2 pada saat memulai program diikuti nama <i>file</i> yang valid	LOAD	Gambar 6.2.1 Gambar 6.2.2	Sesuai yang diharapkan
3	HELP	Menampilkan daftar <i>command</i> dan fungsinya yang dapat dijalankan pada program	Memberikan <i>command</i> HELP atau angka 4 di <i>welcome menu</i> dan <i>login menu</i> , ataupun angka 10 di <i>main menu</i> .	HELP	Gambar 6.3.1 Gambar 6.3.2 Gambar 6.3.3	Sesuai yang diharapkan
4	REGISTER	Memeriksa apakah <i>username</i> bisa didaftarkan tanpa ada duplikasi.	Mengetik REGISTER pada <i>command</i> atau angka 1 setelah START atau LOAD.	REGISTER	Gambar 6.4.1 Gambar 6.4.2	Sesuai yang diharapkan
5	LOGIN	Memeriksa apakah <i>username</i> dan <i>password</i> sudah sesuai dengan data yang dimiliki	Mengetik LOGIN pada <i>command</i> atau angka 2 setelah START atau LOAD.	LOGIN	Gambar 6.5.1 Gambar 6.5.2 Gambar 6.5.3 Gambar 6.5.4	Sesuai yang diharapkan
6	WORK	Menampilkan list pekerjaan dan memeriksa kesesuaian durasi pilihan pekerjaan	Mengetik WORK atau angka 1 pada <i>command</i> setelah berhasil LOGIN, lalu masukkan nomor pekerjaan yang ingin diambil.	WORK	Gambar 6.6.1 Gambar 6.6.2	Sesuai yang diharapkan
7	WORK CHALLENGE	Menampilkan list permainan	Mengetik WORK CHALLENGE atau angka 2 pada <i>command</i>	WORK CHALLENGE	Gambar 6.7.1 Gambar 6.7.2	Sesuai yang diharapkan

		yang dapat dimainkan dan memeriksa apakah pengguna mengalami pengurangan uang setelah masuk dan mengalami peningkatan uang setelah berhasil menang	setelah berhasil LOGIN lalu memasukkan jenis challenge yang ingin dimainkan		Gambar 6.7.1.1 Gambar 6.7.1.2 Gambar 6.7.1.3 Gambar 6.7.2.1 Gambar 6.7.2.2 Gambar 6.7.2.3 Gambar 6.7.2.1 Gambar 6.7.2.2 Gambar 6.7.2.3	
8	BIOWEAPON	Membuat senjata biologis dengan mencocokkan hasil konversi DNA ke protein dengan kode rahasia yang sesuai	Menetik BIOWEAPON atau angka 11 pada <i>command</i> setelah berhasil LOGIN	BIOWEAPON	Gambar 6.8.1 Gambar 6.8.2	Sesuai yang diharapkan
9	STORE LIST	Menampilkan list barang yang terdapat di toko	Menetik STORE LIST atau angka 3 pada <i>command</i> setelah berhasil LOGIN.	STORE LIST	Gambar 6.9.1 Gambar 6.9.2	Sesuai yang diharapkan
10	STORE REQUEST	Memeriksa apakah barang yang di- <i>request</i> berhasil ditambahkan ke antrian	Menetik STORE REQUEST atau angka 4 pada <i>command</i> setelah berhasil LOGIN, lalu masukkan nama barang yang ingin dimasukkan ke dalam antrian	STORE REQUEST	Gambar 6.10.1 Gambar 6.10.2 Gambar 6.10.3	Sesuai yang diharapkan
11	STORE SUPPLY	Memeriksa apakah barang yang ada di antrian berhasil diterima, ditunda, atau ditolak.	Menetik STORE SUPPLY atau angka 5 pada <i>command</i> setelah berhasil LOGIN, lalu masukkan pilihan yang ingin dilakukan kepada barang	STORE SUPPLY	Gambar 6.11.1 Gambar 6.11.2 Gambar 6.11.3 Gambar 6.11.4	Sesuai yang diharapkan

12	STORE REMOVE	Memeriksa apakah barang yang ada di toko berhasil di hapus atau tidak	Menetik STORE REMOVE atau angka 6 pada <i>command</i> setelah berhasil LOGIN, lalu masukkan nama barang yang ingin dihapus dari toko	STORE REMOVE	Gambar 6.12.1 Gambar 6.12.2	Sesuai yang diharapkan
13	LOGOUT	Memeriksa apakah pengguna berhasil keluar dari akun	Menetik LOGOUT atau angka 7 pada <i>command</i> setelah berhasil LOGIN	STORE LOGOUT	Gambar 6.13.1	Sesuai yang diharapkan
14	EXIT	Memberikan pilihan untuk melakukan <i>save file</i> atau keluar dari program	Menetik EXIT atau angka 9 pada <i>command</i> setelah berhasil LOGIN	EXIT	Gambar 6.14.1 Gambar 6.14.2	Sesuai yang diharapkan
15	SAVE	Melakukan <i>save file</i>	Menetik SAVE pada <i>command</i> atau angka 8 setelah berhasil LOGIN, lalu masukkan nama file yang ingin dijadikan tempat penyimpanan	SAVE	Gambar 6.15.1 Gambar 6.15.2 Gambar 6.15.3	Sesuai yang diharapkan

8 Pembagian Kerja dalam Kelompok

<i>Nama Anggota - NIM</i>	<i>Pembagian Kerja</i>
Wisa Ahmaduta Dinutama - 18223003	Membuat ADT Arrayitems dan Arrayusers, memodifikasi ADT Mesinkarakter dan Mesinkata untuk memudahkan implementasi fungsi SAVE dan LOAD, membuat fungsi SAVE dan LOAD, melakukan pengujian program pada bagian SAVE dan LOAD sekaligus memperbaiki <i>bug</i> yang ada, menyunting laporan bagian data test khususnya pada fungsi START, SAVE, dan LOAD.
Carissa Zahrani Putri - 18221093	Membuat ADT Mesinkata, membuat ADT Mesinkarakter, membuat dan testing fungsi STORE LIST, STORE REQUEST, STORE SUPPLY, STORE REMOVE
Catherine Alicia N - 18223069	Membuat HELP, melakukan notulensi pada

	asistensi 1, membuat laporan bagian ringkasan, semua struktur data, program utama, data test start, load, help, register, login, work, store list, store request, store supply, store remove, logout, exit, save, test script, dan lampiran, membuat README pada repository
Muhammad Aqmar Fayyaz Zakaria - 18223043	Membuat fungsi START, LOAD, EXIT, LOGIN, REGISTER, LOGOUT, dan WORK, melakukan <i>merging, testing, debugging</i> , dan <i>editing</i> code dengan seluruh code lainnya dalam main.c dan console.c, dan menyunting laporan bagian ringkasan, program utama, <i>data test</i> , dan <i>test script</i> .
Hugo Benedicto Tanidi - 18221131	Membuat dan menguji WORK CHALLENGE Tebak Angka dan WORDL3, membuat dan menguji bonus Quantum WORDL3 dan Bioweapon, membantu melengkapi laporan terkait <i>work challenge</i> dan bonus

9 Lampiran

9.1 Deskripsi Tugas Besar

Agen Purry sedang menikmati tidur siangnya ketika dia tiba-tiba mendengar *alarm* dari belakang sofa. Suatu pintu rahasia terbuka di bawah dirinya dan ia jatuh ke ruang bawah tanah dan langsung disambut dengan misi terbarunya. ***And indeed it was.*** Setelah pertarungan sengit selama 3 bulan 13 hari 2 jam 47 menit dan 2 detik, suplai senjata dan suplai peralatan yang dimiliki OWCA mulai menipis. Harapan kemenangan OWCA mulai memudar. Tanpa disangka, Agen Purry mengeluarkan senjata rahasia miliknya: menjadi orang Bojongsoang yang memiliki kenalan pegawai Borma. Toko Borma adalah komponen penting yang dapat membawakan kemenangan untuk OWCA pada waktu-waktu kritis ini. Sebab, meskipun Borma terlihat seperti *supermarket* pada umumnya, mereka sebenarnya merupakan pemasok barang-barang perang. Namun terdapat satu masalah kecil, Borma masih beroperasi secara tatap muka dan OWCA tidak memiliki *transport* untuk pergi ke Bojongsoang. Untuk menyelesaikan permasalahan ini, OWCA mengontak tim *programmer* paling andalnya untuk merancang suatu sistem jual beli ke Borma dengan nama **PURRMART! Benar, tim tersebut adalah kalian!** Misi ini akan menantang dan menguji kalian. Namun, dengan kerja tim dan tekad yang kuat, kalian pasti dapat menghadapi tantangan ini.

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. Main Menu

Ketika program pertama kali dijalankan, PURRMART akan memperlihatkan *main menu* yang berisi **welcome menu** dan beberapa *command* yaitu **START**, **LOAD**, dan juga **HELP**.

Setelah itu, program akan memasuki *login menu* yang memiliki command **LOGIN**, **REGISTER**, dan juga **HELP**. Jika pengguna berhasil memasuki kredensial suatu akun, maka mereka akan masuk ke menu selanjutnya.

Main menu menerima masukan berupa *command* yang akan dijelaskan pada bagian berikutnya. Program akan terus menerima *command* sampai diberikan *command* **QUIT** yang berlaku pada seluruh menu.

3. Command

Pengguna dapat memasukkan *command-command* berikut.

a. START

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

b. LOAD <filename>

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

c. LOGIN

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

d. LOGOUT

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

e. REGISTER

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

f. WORK

WORK merupakan *command* yang digunakan pengguna untuk mendapatkan uang. Terdapat sejumlah pekerjaan yang bisa dipilih. Setiap pekerjaan memiliki

waktu tunggu yang berbeda-beda dan dengan nominal pendapatan yang berbeda-beda pula. Selama pengguna sedang bekerja, maka sistem tidak bisa digunakan hingga pekerjaan selesai dilakukan.

g. WORK CHALLENGE

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

a) Tebak Angka

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

b) WORDL3

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

h. STORE LIST

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

i. STORE REQUEST

STORE REQUEST adalah *command* yang digunakan untuk meminta penambahan barang baru ke dalam toko. Barang-barang yang diminta akan disimpan di dalam sebuah antrian dan akan dimasukkan ke toko menggunakan

command selanjutnya. Nama barang yang masuk tidak boleh sama dengan nama barang yang sudah ada di toko atau di antrian.

j. STORE SUPPLY

STORE SUPPLY adalah *command* yang digunakan untuk menambahkan barang baru ke dalam toko berdasarkan antrian permintaan. Barang yang berada pada antrian paling depan akan dimasukan ke toko. Pengguna dapat menerima, menunda, atau menolak permintaan.

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

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

k. STORE REMOVE

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

l. HELP

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

m. SAVE <filename>

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




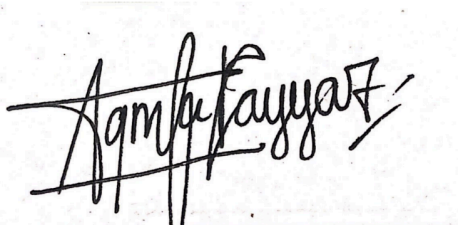
n. QUIT



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

9.2 Notulen Rapat

9.3.1 Notulensi Asistensi 1

Tanggal : 21/11/2024	Catatan Asistensi:
----------------------	--------------------

<p>Tempat : https://meet.google.com/nhq-cccu-fsw</p> <p>Kehadiran Anggota Kelompok:</p> <p>1 18223003</p>  <p>2 18221093</p>  <p>3 18223069</p>  <p>4 18223043</p>  <p>5 18221131</p>	<p>Dijelaskannya aspek-aspek penting dan penjelasan tambahan berdasarkan spesifikasi yang telah diberikan terkait Tugas Besar IF2111 Algoritma dan Struktur Data.</p> <ol style="list-style-type: none"> 1. Dalam program yang kita buat, fitur help tidak perlu dipanggil secara terpisah karena akan langsung ditampilkan saat program dijalankan pertama kali. Namun, menurut masukan dari asisten, fitur help hanya akan muncul jika perintah help dimasukkan secara eksplisit. Oleh karena itu, setelah memulai program dengan perintah start, hanya akan muncul daftar opsi tanpa penjelasan tambahan sebagaimana yang ditampilkan di help. 2. Setelah perintah start dijalankan, konfigurasi default akan langsung dimuat secara otomatis. 3. Perintah exit digunakan sebagai pengganti quit. Hal ini tidak menjadi masalah karena kedua perintah memiliki fungsi yang sama. 4. Dalam contoh yang diberikan, sudah tersedia beberapa akun yang terdaftar. Untuk mempermudah pengujian, file konfigurasi yang digunakan dalam program sebaiknya mengikuti contoh yang telah disediakan dalam spesifikasi tugas.
--	---

	
	<p>Tanda Tangan Asisten:</p> 

9.3 Log Activity Anggota Kelompok

No.	Tanggal	NIM	Nama	Aktivitas
1	15/11/2024	18221093, 18221131, 18223003, 18223043, 18223069	Carissa Zahrani P., Hugo Benedicto T., Wisa Ahmaduta D., M. Aqmar Fayyaz Z., Catherine Alicia N.	Pembagian tugas kelompok
2	17/11/2024	18221093	Carissa Zahrani P.	Membuat ADT Mesinkarakter dan Mesinkata
3	17/11/2024	18221131	Hugo Benedicto T.	Membuat Tebak Angka dan WORDL3
4	20/11/2024	18223003	Wisa Ahmaduta D	Membuat ADT arrayitems dan arrayusers serta memodifikasi ADT mesinkarakter dan mesinkata agar dapat memudahkan implementasi fungsi load dan save.
5	21/11/2024	18221131	Hugo Benedicto T.	Membuat Quantum WORDL3 dan Bioweapon
6	21/11/2024	18223069	Catherine Alicia N	Membuat HELP
7	21/11/2024	18223003	Wisa Ahmaduta Dinutama	Membuat fungsi save dan load.
8	21/11/2024	18221093, 18221131, 18223003, 18223043, 18223069	Carissa Zahrani P., Hugo Benedicto T., Wisa Ahmaduta D., M. Aqmar Fayyaz Z., Catherine Alicia N.	Melakukan asistensi
9	21/11/2024	18223069	Catherine Alicia N	Melakukan notulensi asistensi dan membuat kerangka laporan.

10	22/11/2024	18221093	Carissa Zahrani P.	Membuat fungsi store request, store supply, store list, store remove
11	22/11/2024	18223069	Catherine Alicia	Membuat laporan bagian ringkasan, struktur data, program utama
12	21/11/2024	18223043	M. Aqmar Fayyaz Z.	Membuat kerangka berjalannya program, Membuat fungsi REGISTER, LOGIN, dan LOGOUT serta mengintegrasikannya dengan LOAD dan SAVE ke dalam main.c dan console.c
13	23/11/2024	18223043	M. Aqmar Fayyaz Z.	Mengintegrasikan file main.c dan console.c dengan WORK, STORE LIST, dan STORE REMOVE
14	24/11/2024	18223043	M. Aqmar Fayyaz Z.	Mengintegrasikan file main.c dan console.c dengan STORE REQUEST, STORE SUPPLY, WORK CHALLENGE, dan BIOWEAPON, melakukan penambahan detail detail estetika dalam file, serta melakukan debugging dan testing pada file pra-finishing
15	24/11/2024	18223003	Wisa Ahmaduta D	Melakukan testing pada fitur save dan load sekaligus memperbaiki bug save file yang dapat membuat file berformat

				non-txt dengan memodifikasi fungsi handlesaveonexit.
16	24/11/2024	18223069	Catherine Alicia	Melakukan <i>testing</i> dan mengerjakan laporan bagian <i>Data Test</i> dan <i>Test Script</i>
17	25/11/2024	18223043	M. Aqmar Fayyaz Z.	Melakukan finalisasi code program yang telah disatukan dan melakukan update-update kecil seperti penambahan fitur BACK dan LOGOUT dalam login menu
18	25/11/2024	18223069	Catherine Alici N	Membuat README, melakukan finalisasi laporan untuk semua bagian.