

# LAPORAN TUGAS BESAR

## IF2110 Algoritma dan Struktur Data

### BURBIR


Dipersiapkan oleh:

Kelompok D

1. 13522068 / Adril Putra Merin
2. 13522075 / Marvel Pangondian
3. 13522083 / Evelyn Yosiana
4. 13522092 / Sa'ad Abdul Hakim
5. 13522105 / Fabian Radenta Bangun

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		IF2110-TB-D-02		<jml hlm>
		Revisi	1	24 November 2023

# Daftar Isi

1	Ringkasan	4
2	Penjelasan Tambahan Spesifikasi Tugas	6
2.1	Inisialisasi	6
2.2	Pengguna	6
2.3	Kicauan	6
2.4	Simpan dan Muat	6
3	Struktur Data (ADT)	7
3.1	ADT Sederhana	7
3.2	ADT List dengan Struktur Data Array Statik	14
3.3	ADT Matriks	17
3.4	ADT List dengan Struktur Data Array Dinamik	17
3.5	Mesin Karakter dan Mesin Kata	21
3.6	ADT Priority Queue	24
3.7	ADT Stack	26
3.8	ADT List dengan Struktur Data Berkait	28
3.9	ADT Tree	30
3.10	ADT Graf dengan Representasi Adjacency Matrix	34
4	Program Utama	37
5	Algoritma-Algoritma Menarik	41
5.1	Penghapusan node tree secara cascade.	41
6	Data Test	42
6.1	INISIALISASI	42
6.2	DAFTAR	42
6.3	MASUK	42
6.4	KELUAR	43
6.5	TUTUP_PROGRAM	43
6.6	GANTI_PROFIL	44
6.7	LIHAT_PROFIL	44
6.8	ATUR_JENIS_AKUN	45
6.9	UBAH_FOTO_PROFIL	46
6.10	DAFTAR_TEMAN	47
6.11	HAPUS_TEMAN	48

6.12 TAMBAH_TEMAN	48
6.13 DAFTAR_PERMINTAAN_PERTEMANAN	49
6.14 SETUJUI_PERTEMANAN	50
6.15 KICAU	51
6.16 KICAUAN	51
6.17 SUKA_KICAUAN	52
6.18 UBAH_KICAUAN	53
6.19 BALAS	54
6.20 BALASAN	54
6.21 HAPUS_BALASAN	55
6.22 BUAT_DRAF	56
6.23 LIHAT_DRAF	58
6.24 UTAS	59
6.25 SAMBUNG_UTAS	61
6.26 HAPUS_UTAS	62
6.27 CETAK_UTAS	64
7 Test Script	65
8 Pembagian Kerja dalam Kelompok	75
9 Lampiran	76
9.1 Deskripsi Tugas Besar 1	76
9.2 Notulen Rapat	77
9.3 Log Activity Anggota Kelompok	80

# 1 Ringkasan

Tugas besar mata kuliah algoritma dan struktur data IF2110 tahun 2023 yaitu pembuatan program simulasi bernama Burbir. Program ini dibuat dalam bahasa C berbasis *command-line interface* (CLI). Kompilasi program ini dijalankan pada sistem operasi berbasis UNIX dengan menggunakan Makefile.

Fitur-fitur yang terdapat dalam program ini antara lain inisialisasi, perintah, pengguna, profil, teman, permintaan pertemanan, kicauan, balasan, draf kicauan, utas, simpan dan muat, tagar, kelompok teman, *For Your BurBir* (FYB), serta nomor HP. Fitur-fitur tersebut diimplementasikan dengan memanfaatkan *Abstract Data Type* (ADT) yang telah dipelajari selama satu semester dalam mata kuliah ini. ADT yang kami gunakan dalam program ini antara lain:

- ADT sederhana untuk time dan datetime.
- ADT list dengan struktur data array statik untuk menyimpan daftar pengguna.
- ADT matriks untuk mengimplementasikan foto profil.
- ADT list dengan struktur data array dinamik untuk menyimpan kicauan.
- Mesin karakter dan mesin kata untuk membaca masukan dari pengguna;
- ADT priority queue untuk mengimplementasikan fungsi permintaan pertemanan;
- ADT stack untuk mengimplementasikan fungsi draft;
- ADT list dengan struktur data berkait untuk mengimplementasikan fungsi utas;
- ADT tree untuk mengimplementasikan fungsi balasan;
- ADT graf dengan representasi *adjacency matrix* untuk mengimplementasikan graf pertemanan.

Secara garis besar, laporan ini berisi tentang ringkasan singkat program, penjelasan mengenai struktur data yang digunakan beserta alasan pemilihan struktur data tersebut, penjelasan mengenai program utama, serta hasil pengetesan. Di dalam laporan ini juga terdapat pembagian kerja kelompok, notulen rapat. dan log activity.

Kesimpulan dari program yang telah kami buat yaitu secara garis besar program ini berjalan dengan baik. ADT ADT dari praktikum yang telah kami jalani berhasil terimplementasikan dengan baik dalam program ini.

## 2 Penjelasan Tambahan Spesifikasi Tugas

### 2.1 Inisialisasi

Ketika user memulai program, user terlebih dahulu melakukan load dengan memasukan folder konfigurasi. Setelah berhasil *load* folder, user memiliki lima opsi, yakni masuk, daftar, tutup\_program, simpan, dan muat. User dapat menutup program kapanpun, tetapi harus masuk terlebih dahulu sebelum dapat menggunakan fitur-fitur BurBir.

### 2.2 Pengguna

Pengguna pada BurBir menyimpan beberapa hal yakni nama user, password, nomor telepon, bio, weton, tipe akun, foto profil, daftar pertemanan, serta jumlah teman. Dalam menyimpan pengguna-pengguna pada aplikasi BurBir, terdapat sebuah *array of user* yang statik (20 elemen, sesuai dengan spesifikasi).

### 2.3 Kicauan

Kicauan merupakan sebuah fitur pesan singkat pada aplikasi BurBir. Setiap pengguna dapat membuat, membalas, serta mencari kicau. Kicau memiliki beberapa fitur lain yakni membuat utas dan draf. Utas merupakan rangkaian pesan dengan sebuah kicau sebagai kepalanya (kicauan utama), pengguna dapat membuat, menyambung, menghapus, dan mencetak utas. Pengguna hanya dapat membuat utas pada kicau pengguna sendiri, tidak pada kicau lain. Fitur lain yang terdapat pada kicauan adalah fitur membalas. Pengguna dapat membalas kicau pengguna lain dengan syarat bahwa pengguna lain tersebut memiliki akun publik, atau pengguna tersebut memiliki akun privat tapi berteman dengan pengguna yang ingin membalas.

### 2.4 Simpan dan Muat

Fitur lain yang dapat pengguna lakukan pada aplikasi BurBir adalah fitur simpan dan muat. Simpan merupakan fitur menyimpan data BurBir pengguna kepada sebuah folder config, sedangkan muat merupakan fitur yang membaca sebuah folder config untuk mempersiapkan data yang akan digunakan.

### 3 Struktur Data (ADT)

#### 3.1 ADT Sederhana

Penjelasan sketsa struktur data: terdapat struktur data TIME, DATETIME, dan pii. struktur data TIME terdiri atas HH (menunjukkan jam), MM (menunjukkan menit), dan SS (menunjukkan detik). HH bertipe integer dengan range 0 - 23, MM bertipe integer dengan range nol sampai 0 - 59, sedangkan SS bertipe integer dengan range 0 - 59. Struktur data DATETIME terdiri atas DD (menunjukkan tanggal), MM (menunjukkan bulan), YYYY (menunjukkan tahun), dan T. DD, MM, dan YYYY bertipe integer sedangkan T bertipe TIME. DD berada pada range 1 - 31, MM pada range 1 - 12, sedangkan YYYY pada range 1900 - 2030. Struktur data pii terdiri atas first dan second yang masing-masing bertipe integer. Pii merupakan singkatan dari pair of integer yang akan digunakan pada ADT priority queue.

Prototipe dan primitif pada ADT TIME antara lain:

- konstruktor untuk membentuk TIME;
- predikat untuk mengecek kevalidan TIME serta mengecek hubungan antara TIME 1 dengan TIME 2 apakah sama, tidak sama, lebih kecil dari, atau lebih besar dari;
- primitif lain untuk membaca dan menulis TIME, mengubah TIME menjadi detik dan sebaliknya, serta mengirim satu atau N detik setelah maupun sebelum TIME.

Prototipe dan primitif pada ADT DATETIME antara lain:

- konstruktor untuk membentuk DATETIME;
- selektor untuk mendapatkan jumlah hari maksimum pada bulan dan tahun tertentu, mendapatkan durasi dari dua DATETIME;
- predikat untuk mengecek kevalidan DATETIME, mengecek hubungan antara DATETIME 1 dengan DATETIME 2 apakah sama, tidak sama, lebih kecil dari, atau lebih besar dari;
- primitif lain untuk membaca dan menulis DATETIME, mengirimkan salinan D dengan detik ditambah maupun dikurang dengan N detik, mengubah array of character menjadi DATETIME, serta menghasilkan Word hasil DATETIME.

Persoalan yang diselesaikan: pencetakan waktu dan tanggal kicauan, balasan, dan utas ketika diterbitkan dari waktu lokal.

Alasan pemilihan: karena berguna untuk pemformatan waktu yang didapat saat pembuatan kicauan, balasan, dan sebagainya dari time.h.

Diimplementasikan sebagai ADT TIME dengan file header “time.h” dan ADT DATETIME dengan file header “datetime.h”.

Berikut header dari ADT TIME.

```
/* File: time.h */
/* Tanggal: 3 September 2022 */
/* Definisi ADT TIME */

#ifndef TIME_H
#define TIME_H

#include "../boolean.h"
#include <time.h>

/* *** Definisi TYPE TIME <HH:MM:SS> *** */
typedef struct
{
    int HH; /* integer [0..23] */
    int MM; /* integer [0..59] */
    int SS; /* integer [0..59] */
} TIME;

/* *** Notasi Akses: selektor TIME *** */
#define Hour(T) (T).HH
#define Minute(T) (T).MM
#define Second(T) (T).SS

boolean IsTIMEValid(int H, int M, int S);
/* Mengirim true jika H,M,S dapat membentuk T yang valid */
/* dipakai untuk mentest SEBELUM membentuk sebuah Jam */
```

```

/* *** Konstruktor: Membentuk sebuah TIME dari komponen-komponennya
*** */
void CreateTime(TIME *T, int HH, int MM, int SS);
/* Membentuk sebuah TIME dari komponen-komponennya yang valid */
/* Prekondisi : HH, MM, SS valid untuk membentuk TIME */

void BacaTIME(TIME *T);
/* I.S. : T tidak terdefinisi */
/* F.S. : T terdefinisi dan merupakan jam yang valid */
/* Proses : mengulangi membaca komponen HH, MM, SS sehingga membentuk
T */
/* yang valid. Tidak mungkin menghasilkan T yang tidak valid. */
/* Pembacaan dilakukan dengan mengetikkan komponen HH, MM, SS
dalam satu baris, masing-masing dipisahkan 1 spasi, diakhiri
enter. */
/* Jika TIME tidak valid maka diberikan pesan: "Jam tidak valid", dan
pembacaan
diulangi hingga didapatkan jam yang valid. */
/* Contoh:
60 3 4
Jam tidak valid
1 3 4
--> akan terbentuk TIME <1,3,4> */

void TulisTIME(TIME T);
/* I.S. : T sembarang */
/* F.S. : Nilai T ditulis dg format HH:MM:SS */
/* Proses : menulis nilai setiap komponen T ke layar dalam format
HH:MM:SS
tanpa karakter apa pun di depan atau belakangnya, termasuk spasi,
enter, dll.*/

long TIMEToDetik(TIME T);

```



```

/* Diberikan sebuah TIME, mengkonversi menjadi jumlah detik dari
pukul 0:0:0 */
/* Rumus : detik = 3600*HH + 60*MM + SS */
/* Nilai maksimum = 3600*23+59*60+59 */

TIME DetikToTIME(long N);
/* Mengirim konversi detik ke TIME */
/* Catatan: Jika N >= 86400, maka harus dikonversi dulu menjadi
jumlah detik yang
    mewakili jumlah detik yang mungkin dalam 1 hari, yaitu dengan
rumus:
    N1 = N mod 86400, baru N1 dikonversi menjadi TIME */

/* *** Kelompok Operator Relational *** */
boolean TEQ(TIME T1, TIME T2);
/* Mengirimkan true jika T1=T2, false jika tidak */
boolean TNEQ(TIME T1, TIME T2);
/* Mengirimkan true jika T1 tidak sama dengan T2 */
boolean TLT(TIME T1, TIME T2);
/* Mengirimkan true jika T1<T2, false jika tidak */
boolean TGT(TIME T1, TIME T2);
/* Mengirimkan true jika T1>T2, false jika tidak */
/* *** Operator aritmatika TIME *** */
TIME NextDetik(TIME T);
/* Mengirim 1 detik setelah T dalam bentuk TIME */
TIME NextNDetik(TIME T, int N);
/* Mengirim N detik setelah T dalam bentuk TIME */
TIME PrevDetik(TIME T);
/* Mengirim 1 detik sebelum T dalam bentuk TIME */
TIME PrevNDetik(TIME T, int N);
/* Mengirim N detik sebelum T dalam bentuk TIME */
/* *** Kelompok Operator Aritmetika *** */
long Durasi(TIME TAw, TIME Takh);
/* Mengirim Takh-TAw dlm Detik, dengan kalkulasi */

```

```

/* Jika TAw > Takh, maka Takh adalah 1 hari setelah TAw */

#endif

```

Berikut header dari ADT DATETIME.

```

/* File: datetime.h */
/* Tanggal: 30 Agustus 2023 */
/* Definisi ADT DATETIME */

#ifndef DATETIME_H
#define DATETIME_H

#include "../boolean.h"
#include "../time/time.h"
#include "../wordmachine/wordmachine.h"

/* *** Definisi TYPE DATETIME <DD/MM/YY HH:MM:SS> *** */
typedef struct
{
    int DD; /* integer [1..31] */
    int MM; /* integer [1..12] */
    int YYYY; /* integer [1900..2030] */
    TIME T;
} DATETIME;

/* *** Notasi Akses: selektor DATETIME *** */
#define Day(D) (D).DD
#define Month(D) (D).MM
#define Year(D) (D).YYYY
#define Time(D) (D).T

int GetMaxDay(int M, int Y);

/* Mengirimkan jumlah hari maksimum pada bulan M dan tahun Y */
/* Prekondisi: 1 <= M <= 12 */

```

```

/* Hint: Perhatikan Leap Year. Leap Year adalah tahun dengan 29 hari
pada bulan Februari */
/* Aturan Leap Year: */
/* 1. Jika angka tahun itu habis dibagi 400, maka tahun itu sudah
pasti tahun kabisat. 8*/
/* 2. Jika angka tahun itu tidak habis dibagi 400 tetapi habis dibagi
100, maka tahun itu sudah pasti bukan merupakan tahun kabisat. */
/* 3. Jika angka tahun itu tidak habis dibagi 400, tidak habis dibagi
100 akan tetapi habis dibagi 4, maka tahun itu merupakan tahun
kabisat. */
/* 4. Jika angka tahun tidak habis dibagi 400, tidak habis dibagi
100, dan tidak habis dibagi 4, maka tahun tersebut bukan merupakan
tahun kabisat. */

boolean IsDATETIMEValid(int D, int M, int Y, int h, int m, int s);
/* Mengirim true jika D,M,Y,h,m,s dapat membentuk D yang valid */
/* dipakai untuk mentest SEBELUM membentuk sebuah DATETIME */

/* *** Konstruktor: Membentuk sebuah DATETIME dari
komponen-komponennya *** */
void CreateDATETIME(DATETIME *D, int DD, int MM, int YYYY, int hh,
int mm, int ss);
/* Membentuk sebuah DATETIME dari komponen-komponennya yang valid */
/* Prekondisi : DD, MM, YYYY, h, m, s valid untuk membentuk DATETIME
*/

void BacaDATETIME(DATETIME *D);
/* I.S. : D tidak terdefinisi */
/* F.S. : D terdefinisi dan merupakan DATETIME yang valid */
/* Proses : mengulangi membaca komponen DD, MM, YY, h, m, s sehingga
membentuk D */
/* yang valid. Tidak mungkin menghasilkan D yang tidak valid. */
/* Pembacaan dilakukan dengan mengetikkan komponen DD, MM, YY, h, m,
s

```

```

    dalam satu baris, masing-masing dipisahkan 1 spasi, diakhiri
    enter. */
/* Jika DATETIME tidak valid maka diberikan pesan: "DATETIME tidak
    valid", dan pembacaan
    diulangi hingga didapatkan DATETIME yang valid. */
/* Contoh:
    32 13 2023 12 34 56
    DATETIME tidak valid
    31 12 2023 12 34 56
    --> akan terbentuk DATETIME <31,12,2023,12,34,56> */

void TulisDATETIME(DATETIME D);
/* I.S. : D sembarang */
/* F.S. : Nilai D ditulis dg format DD/MM/YYYY HH:MM:SS */
/* Proses : menulis nilai setiap komponen D ke layar dalam format
    DD/MM/YYYY HH:MM:SS
    tanpa karakter apa pun di depan atau belakangnya, termasuk spasi,
    enter, dll.*/

/* *** Kelompok operasi relasional terhadap DATETIME *** */
boolean DEQ(DATETIME D1, DATETIME D2);
/* Mengirimkan true jika D1=D2, false jika tidak */
boolean DNEQ(DATETIME D1, DATETIME D2);
/* Mengirimkan true jika D1 tidak sama dengan D2 */
boolean DLT(DATETIME D1, DATETIME D2);
/* Mengirimkan true jika D1<D2, false jika tidak */
boolean DGT(DATETIME D1, DATETIME D2);
/* Mengirimkan true jika D1>D2, false jika tidak */
DATETIME DATETIMENextNDetik(DATETIME D, int N);
/* Mengirim salinan D dengan detik ditambah N */
DATETIME DATETIMEPrevNDetik(DATETIME D, int N);
/* Mengirim salinan D dengan detik dikurang N */
/* *** Kelompok Operator Aritmetika terhadap DATETIME *** */
long int DATETIMEDurasi(DATETIME DAw, DATETIME DAkh);

```

```

/* Mengirim DAKh-DAw dlm Detik, dengan kalkulasi */
/* Prekondisi: DAKh > DAw */

DATETIME string_toDate_time(Word date_insert);
/*menerima array of characters, dan membentuk datetime*/

Word dateTimeToWord(DATETIME t);
// mengeluarkan Word hasil datetime

#endif

```

Berikut header dari ADT pii.

```

#ifndef PII_H
#define PII_H
typedef struct
{
    int first;
    int second;
} pii; // pair of integer

#endif

```

## 3.2 ADT List dengan Struktur Data Array Statik

Penjelasan sketsa struktur data: struktur data listUser terdiri atas listU yang merupakan list yang berisi pengguna aplikasi BurBir serta Neff yang merupakan banyaknya pengguna yang terdaftar pada aplikasi BurBir. listU bertipe User, sedangkan Neff bertipe integer.

Prototipe dan primitif pada ADT ini antara lain:

- konstruktor untuk membuat listUser;
- selektor untuk mencari user dengan id;
- predikat untuk mengecek kevalidan password dan user;

- primitf lainnya untuk sign up, login, menambah dan menghapus teman, serta menambah dan membatalkan permintaan pertemanan.

Persoalan yang diselesaikan: penyimpanan daftar teman.

Alasan pemilihan: karena jumlah elemen maksimum/pengguna yang kecil (20) sehingga memori tidak perlu dialokasikan secara dinamis.

Diimplementasikan sebagai ADT list dengan struktur data array statik dengan file header "listuser.h".

Berikut header dari ADT list dengan struktur data array statik.

```
#ifndef LISTUSER_H
#define LISTUSER_H
#include <stdio.h>
#include "user.h"
#include "../friend/friend.h"
#include "../friend/friendrequest.h"
#include "../../lib/boolean.h"
#include "../../lib/charmachine/charmachine.h"
#include "../../lib/wordmachine/wordmachine.h"
#include "../../lib/dsu/dsu.h"

typedef struct
{
    User listU[20];
    int Neff;
}ListUser;

// CONSTRUCTOR
void CreateListUser(ListUser * listuser);

// SELECTOR
int searchUserByID(ListUser listuser, Word name);

// AUTHENTICATION
void SignUp(ListUser *listuser, int *currIdx);
```

```

void LogIn(ListUser *listuser, int *currIdx);

int isUsernameNotValid(Word name, ListUser listuser);
/*
    code 0 : valid
    code 1 : invalid word length
    code 2 : invalid username (already used)
    code 3 : name consist of whitespaces only
*/

boolean isPasswordValid(Word password);

// FRIEND RELATED FUNCTION
void friendList(ListUser listuser, int currIdx, Friend friend);

void deleteFriend(ListUser *listuser, int currIdx, Friend *friend);

void addFriendReq(ListUser *listuser, int currIdx, Friend *friend);

void cancelFriendReq(ListUser *listuser, int currIdx, Friend
*friend);

void displayFriendRequestList(ListUser listuser, int currIdx);

void confirmFriendRequest(ListUser *listuser, int currIdx, Friend
*friend);

// FRIEND GROUP
void searchFriendGroup(ListUser listuser, int currIdx, Friend
friend);

#endif

```

### 3.3 ADT Matriks

Penjelasan sketsa struktur data: struktur Matrix terdiri atas array of array character dengan nama Buffer. Matriks dibatasi dengan ukuran  $5 \times 10$  dengan kolom ganjil merepresentasikan bentuk dan kolom genap merepresentasikan warna.

Prototipe dan primitif pada ADT ini antara lain:

- konstruktor untuk membentuk matriks.

Persoalan yang diselesaikan: menampilkan profile picture pengguna.

Alasan pemilihan: karena profile picture pengguna bisa dipresentasikan lebih mudah ke dalam sebuah matriks.

Diimplementasikan sebagai ADT matriks dengan file header “matrix.h”

Berikut header dari ADT matriks.

```
#ifndef MATRIX_H
#define MATRIX_H
#include <stdio.h>
#include "../boolean.h"
typedef struct matrix
{
    char Buffer[5][10];
} Matrix;

void CreateMatrix(Matrix *mat);

#endif
```

### 3.4 ADT List dengan Struktur Data Array Dinamik

Penjelasan sketsa struktur data: struktur data listKicauan terdiri atas pointer to buffer serta neff dan capacity yang bertipe integer. Buffer sendiri bertipe bentukan ElType (Kicauan) yang terdiri atas id, like, idUtas dan idAuthor yang bertipe integer, text, tag, dan author



yang bertipe Word, time yang bertipe DATETIME, ut yang bertipe ListUtas, serta treebalasan yang bertipe AdressTree.

Prototipe dan primitif pada ADT ini antara lain:

- konstruktor untuk membuat dan mendealokasi listKicauan;
- selektor untuk menambah dan mengubah kicauan;
- predikat untuk mengecek apakah list penuh dan apakah pengguna memiliki kicauan.
- primitf untuk menampilkan ListKicauan, menduplikat ListKicauan, menambah kapasitas ListKicauan, memberikan like pada kicauan, mencetak utas, mencari HashTag, serta menyambung dan memutus utas.

Persoalan yang diselesaikan: penyimpanan dan modifikasi kicauan.

Alasan pemilihan: Pengguna dapat menambah kicauan, jadi jumlah kicauan akan terus bertambah atau berkurang. Dipilihnya struktur data array dinamik untuk mengatasi jumlah kicauan yang berganti-ganti tersebut.

Diimplementasikan sebagai ADT list dengan file header 'listkicauan.h'.

Berikut header dari ADT ListKicauan dengan struktur data array dinamik.

```
#ifndef __LISTKICAUAN_H__
#define __LISTKICAUAN_H__
#include <stdio.h>
#include <stdlib.h>
#include "tweet.h"
#include "../../lib/boolean.h"
#include "../../lib/charmachine/charmachine.h"
#include "../../lib/wordmachine/wordmachine.h"
#include "../hashtag/hashtag.h"
#include "../friend/friend.h"
#include "../user/listuser.h"
/* Indeks minimum list */
#define IDX_MIN 0

/* Indeks tak terdefinisi*/

/* Definisi elemen dan koleksi objek */
```

```

typedef Kicauan ElType; /* type elemen list */
typedef int IdxType;

typedef struct listKicauan
{
    ElType *buffer;
    int nEff;
    int capacity;
} ListKicauan;

#define NEFF(l) (l).nEff
#define BUFFER(l) (l).buffer
#define ELMT(l, i) (l).buffer[i]
#define CAPACITY(l) (l).capacity

void createListKicauan(ListKicauan *l, int capacity);

void dealocateListKicauan(ListKicauan *l);

void bacaKicauan(ListKicauan *lkic, User currUser, int *IdKicau, int
currId, HashTag * hashtag);

void displayListKicauan(ListKicauan lkic, Friend friend, int currId);

void copyListKicauan(ListKicauan lIn, ListKicauan *lOut);

void expandListKicauan(ListKicauan *l, int num);

boolean isFullListKicauan(ListKicauan l);

void likeKicau(ListKicauan *lkic, ListUser l, Friend friend, int
idKicauYangInginDiLike, int currIdx);

```

```

void updateKicau(ListKicauan *lkic, int currIdx, int idKicau);

boolean userOwnsKicau(ListKicauan lkic, int currIdx, int idKicau);

void makeKicauanUtama(ListKicauan *lkic, int currIdx, int idKicau,
int *idUtas);

void cetakUtas(ListKicauan lkic, Friend friend, ListUser lUser, int
currIdx, int idUtas);

void sambungUtas(ListKicauan *lkic, int currIdx, int idUtas, int
index);

void putusUtas(ListKicauan *lkic, DATETIME *date, Word *text, int
currIdx, int idUtas, int index);

void searchHashtag(ListKicauan lkic, HashTag *hashtag, Word tag, int
currID, Friend friendGraph, ListUser listuser);
#endif

```

```

#ifndef __TWEET_H__
#define __TWEET_H__

typedef struct kicauan
{
    int id;
    Word text;
    int like;
    Word author;
    DATETIME time;
    ListUtas ut; // null jika tidak ada
    int idUtas; // -1 jika tidak ada
    int idAuthor; // lokasi user di listuser, jadi currId

```

```

    Word tag;
    // pointer to tree

} Kicauan;
#define ID(kic) (kic).id
#define TEXTKICAU(kic) (kic).text
#define LIKE(kic) (kic).like
#define AUTHOR(kic) (kic).author
#define DATEKICAU(kic) (kic).time
#define UTAS(kic) (kic).ut
#define IDUTAS(kic) (kic).idUtas
#define IDAUTHOR(kic) (kic).idAuthor
#define TAG(kic) (kic).tag

void createKicauan(int id, Word text, int like, Word author, DATETIME
time, Kicauan *kic, int idAuthor, Word tag);

// KICAUAN , harus ada friendgroup dulu
void ubahKicau(int idKicau);

void displayKicau(Kicauan kic); // display sebuah kicau

#endif

```

### 3.5 Mesin Karakter dan Mesin Kata

Penjelasan sketsa struktur data: struktur data Word terdiri atas TabWord yang merupakan array of character dan Length bertipe integer yang menyimpan panjang Word. Spasi (' ') didefinisikan sebagai BLANK. newline (\n) sebagai penanda akhir kalimat untuk kemudian berlanjut ke baris selanjutnya didefinisikan sebagai SPACE. Carriage (r) sebagai penanda akhir kalimat tanpa berlanjut ke baris selanjutnya didefinisikan sebagai CARRIAGE.

Prototipe yang terdapat pada mesin karakter terdiri atas prosedur-prosedur dasar untuk mulai mengakuisisi karakter dan berpindah ke karakter selanjutnya. Sedangkan prototipe yang terdapat pada mesin kata terdiri atas prosedur-prosedur dasar untuk mulai mengakuisisi kata, mengabaikan BLANK, berganti ke kata selanjutnya, meng-*copy* kata dan membuat kata. Terdapat juga prosedur-prosedur spesifik, antara lain membaca command, menampilkan kata dengan maupun tanpa enter. Selain itu, ada juga fungsi untuk mengubah kata menjadi integer serta predikat untuk membandingkan kata maupun karakter dan mengecek apakah seluruh kata dalam kalimat merupakan spasi atau bukan.

Persoalan yang diselesaikan: Membaca command dari pengguna dan data dari file config.

Alasan pemilihan: karena mesin kata dan mesin karakter dapat membaca masukan dari terminal.

ADT ini diimplementasikan sebagai mesin karakter dengan file header “charmachine.h” dan mesin kata dengan file header ‘wordmachine.h’

Berikut header dari mesin karakter.

```
#ifndef __CHAR_MACHINE_H__
#define __CHAR_MACHINE_H__

#include "../boolean.h"

#define MARK ';'

extern char currentChar;
extern boolean EOP;

void START();

void ADV();

#endif
```

Berikut header dari mesin kata.

```
#ifndef __WORDMACHINE_H__
```

```

#define __WORDMACHINE_H__

#define NMax 1024
#define BLANK ' '
#define SPACE '\n'
#define CARRIAGE '\r'
typedef struct
{
    char TabWord[NMax]; /* container penyimpan kata, indeks yang
dipakai [0..NMax-1] */
    int Length;
} Word;

extern boolean EndWord;
extern Word currentWord;

void IgnoreBlanks();

void STARTWORD();

void ADVWORD();

void CopyWord();

void CreateWord(Word *w);

Word readWord(int len);

void readCommand(Word *command);

void displayWord(Word w);

void displayWordWithoutEnter(Word w);

```

```

boolean isWordEqual(Word w1, Word w2); // case sensitive

boolean isCharEqual(Word w1, Word w2); // case insensitive

void assignWord(Word *w, char arr[], int len);

int wordToInteger(Word w);

boolean isAllSpace(Word w);

#endif

```

### 3.6 ADT Priority Queue

Penjelasan sketsa struktur data: terdapat struktur data PQNode yang terdiri atas info bertipe pii dan next yang bertipe PQAddress. Pii sendiri merupakan struktur data bentukan yang terdiri atas first dan second yang masing-masing bertipe integer.

Prototipe dan primitif pada ADT ini antara lain:

- konstruktor untuk membentuk PQAddress baru
- selektor untuk mendapatkan panjang dari priority queue.
- predikat untuk mengecek apakah priority queue kosong atau tidak.
- primitf untuk menampilkan priority queue, serta melakukan enqueue dan dequeue.

Persoalan yang diselesaikan: penyimpanan dan pengelolaan permintaan pertemanan.

Alasan pemilihan: karena ADT priority queue dapat mengurutkan pengguna dari yang paling banyak memiliki teman.

Diimplementasikan sebagai ADT priority queue dengan file header 'priorityqueue.h'.

Berikut header dari priority queue.

```

#ifndef PRIOQUEUE_H
#define PRIOQUEUE_H

typedef struct pqnode *PQAddress;

```

```

typedef struct pqnode
{
    pii info;
    PQAddress next;
} PQNode;

typedef PQAddress PriorityQueue;

#define NEXT(p) (p)->next
#define INFO(p) (p)->info

#define FRONT(pq) (pq)->info

/* Constructor */
PQAddress newPQAddress(pii val);
void CreatePriorityQueue(PriorityQueue *pq);
void DisplayPriorityQueue(PriorityQueue pq);

/* Priority Queue Properties */
int PQLength(PriorityQueue pq);
boolean isPQEmpty(PriorityQueue pq);

/* Enqueue/Dequeue Primitive */
void enqueue(PriorityQueue *pq, pii val);
void dequeue(PriorityQueue *pq, pii *val);

#endif

```

```

#ifndef PII_H
#define PII_H
typedef struct
{
    int first;
    int second;

```



```

} pii; // pair of integer

#endif

```

### 3.7 ADT Stack

Penjelasan sketsa struktur data: terdapat struktur data StackDraft yang terdiri atas addrTopDraft bertipe Address, struktur data Node yang terdiri atas info bertipe Draft dan next bertipe Address, serta struktur data Draft yang terdiri atas word dan tag bertipe Word serta time bertipe DATETIME.

Prototipe dan primitif pada ADT ini antara lain:

- konstruktor untuk membentuk draft dan stack draft, serta mendealokasi draft;
- predikat untuk mengecek apakah draft kosong atau tidak;
- primitif untuk menampilkan draft, mem-*publish* draft, membuat draf baru, dan melakukan push pada stack draft.

Persoalan yang diselesaikan: Penyimpanan dan pengelolaan draft.

Alasan pemilihan : Untuk menyelesaikan persoalan program draf, digunakan ADT Stack karena draf yang dapat diubah, dihapus, ataupun diterbitkan oleh pengguna hanyalah draf yang terakhir dibuat. Konsep ini sangat cocok dengan implementasi ADT stack yang hanya dapat mengakses elemen yang paling terakhir dimasukkan. Stack yang diaplikasikan pada program adalah stack dengan struktur data berkait karena draf yang dimasukkan oleh pengguna tidak dibatasi jumlahnya, sehingga tidak diketahui berapa jumlah maksimal elemen stack yang dapat diterima. Stack dengan struktur data berkait dapat menangani kasus ini karena ia tidak terbatas oleh jumlah elemen tertentu.

Diimplementasikan sebagai ADT stack dengan file header 'liststackdraft.h'.

Berikut header dari ADT list stack draf.

```

#ifndef __LISTDRAFT_H__
#define __LISTDRAFT_H__

#define MaxElListStack 20

```

```

typedef StackDraft ListElType;
typedef struct liststackdraft
{
    ListElType contents[MaxElListStack];
} ListStackDraft;

void assignStackDraft(ListStackDraft *l, int userID, StackDraft sd);

void CreateListStackDraft(ListStackDraft *l);

#endif

```

```

#ifndef __DRAFT_H__
#define __DRAFT_H__

typedef struct draft
{
    Word word;
    Word tag;
    DATETIME time;
} Draft;

typedef struct node *Address;
typedef struct node
{
    Draft info;
    Address next;
} Node;

typedef struct
{
    Address addrTopDraft;
} StackDraft;

```

```

#define NEXT(p) (p)->next
#define INFO(p) (p)->info
#define ADDR_TOP(sd) (sd).addrTopDraft
#define TOP(sd) (sd).addrTopDraft->info
#define WORD(d) (d).word
#define TIME(d) (d).time

Address newNodeDraft(Draft x);

void CreateDraft(Draft *d, Word w, DATETIME time, Word tag);

void CreateStackDraft(StackDraft *sd);

void pushDraft(StackDraft *sd, Draft d);

void deleteDraft(StackDraft *sd, Draft *d);

boolean isEmptyDraft(StackDraft sd);

void seeDraft(ListKicauan *lkic, int currIdx, Word authorName,
StackDraft *sd, int *IdKicau, HashTag *hashtag);

void publishDraft(int id_kicau, int id_user, Word user_name, Kicauan
*k, Draft d, Word author, int idAuthor);

void makeDraft(ListKicauan *lkic, int currIdx, Word authorName,
StackDraft *st, int *IdKicau, HashTag *hashtag);

#endif

```

### 3.8 ADT List dengan Struktur Data Berkait

Penjelasan sketsa struktur data: struktur data thread terdiri atas bentukan DATETIME, WORD, serta AddressUtas yang merupakan pointer ke utas berikutnya

STEI- ITB	IF2110-TB-D-02	Halaman 27 dari 82 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.		

Prototipe dan primitif pada ADT ini antara lain:

- konstruktor untuk membuat dan mendealokasi Utas;
- selektor untuk menambah serta menghapus Utas;
- predikat untuk mengecek apakah utas kosong;
- primitif untuk menampilkan Utas.

Persoalan yang diselesaikan: penyimpanan dan modifikasi utas

Alasan pemilihan: utas merupakan rangkaian pesan yang saling berkait dengan satu sama lain, sehingga dipilihnya ADT List dengan Struktur Data Berkait untuk mengikuti sifat utas tersebut.

Diimplementasikan sebagai ADT List dengan Struktur Data Berkait dengan file header 'thread.h'

Berikut header dari list dengan struktur data berikut:

```
#ifndef __THREAD_H__
#define __THREAD_H__

typedef struct utas *AddressUtas;
typedef struct utas
{
    DATETIME date;
    Word text;
    AddressUtas nextUtas;
} Utas;
typedef AddressUtas ListUtas;

#define DATEUTAS(p) (p)->date
#define TEXTUTAS(p) (p)->text
#define NEXTUTAS(p) (p)->nextUtas

AddressUtas newNode(DATETIME date, Word text);

#define IDX_UNDEF (-1)
```

```

#define FIRST(l) (l)

void CreateUtas(ListUtas *l); // membentuk utas kosong

boolean isEmptyUtas(ListUtas l);

void insertFirstListUtas(ListUtas *l, DATETIME date, Word text);

void insertLastListUtas(ListUtas *l, DATETIME date, Word text);

void displayListUtas(ListUtas l, Word author);

void insertAtListUtas(ListUtas *l, DATETIME date, Word text, int
index);

void deleteFirstListUtas(ListUtas *l, DATETIME *date, Word *text);

void deleteLastListUtas(ListUtas *l, DATETIME *date, Word *text);

void deleteAtListUtas(ListUtas *l, int idx, DATETIME *date, Word
*text);

#endif

```

### 3.9 ADT Tree

Penjelasan sketsa struktur data: terdapat struktur data Balasan, NodeBalasan, dan TreeBalasan. NodeBalasan yang terdiri atas struktur data info, left, dan right. Left dan right masing-masing merupakan struktur data bertipe pointer to NodeBalasan, sedangkan info berupa struktur data Balasan yang terdiri atas id, text, author, time, dan idAuthorBalasan. id dan idAuthorBalasan bertipe integer, text dan author bertipe word, sedangkan time bertipe DATETIME. Struktur TreeBalasan terdiri atas parentNode bertipe AddressTree dan nEff bertipe integer. AddressTree sendiri merupakan pointer to nodeBalasan.

Prototipe dan primitif pada ADT ini antara lain:

- konstruktor untuk membentuk dan mendealokasi node, membentuk tree kosong, dan menghapus tree;
- selektor untuk mendapatkan address dari id node;
- predikat untuk mengecek apakah tree kosong, tree satu elemen, serta mengecek apakah node memiliki left atau/dan right; serta
- primitf untuk menambahkan node pada tree dan mendapatkan address dari id node.

Persoalan yang diselesaikan: penyimpanan balasan dari kicauan.

Alasan pemilihan: tree cocok memodelkan hubungan antarbalasan, dimana dalam program ini, right child dari suatu node mengindikasikan adanya kesetaraan (kedua node merupakan balasan dari node yang sama), sedangkan left child dari suatu node merupakan balasan dari node tersebut. Selain itu, pemeliharaan tree untuk balasan relatif mudah, termasuk untuk penambahan dan penghapusan node balasan. tree yang dipakai dalam program ini merupakan binary tree yang memungkinkan untuk optimalisasi memori.

Diimplementasikan sebagai ADT binary tree dengan file header “treebalasan.h”

Berikut implementasi dari file “treebalasan.h” dan “reply.h”.

```
#ifndef __REPLY_H__
#define __REPLY_H__

#include "../lib/datetime/datetime.h"
#include "../lib/wordmachine/wordmachine.h"
#include "../lib/datetime/datetime.h"
#include <stdio.h>
#include <stdlib.h>

typedef struct balasan
{
    int id;
    Word text;
    Word author;
    DATETIME time;
    int idAuthorBalasan;
} Balasan;
```

```

#define IDBALASAN(balasan) (balasan).id
#define TEXTBALASAN(balasan) (balasan).text
#define AUTHORBALASAN(balasan) (balasan).author
#define DATEBALASAN(balasan) (balasan).time
#define IDAUTHORBALASAN(balasan) (balasan).idAuthorBalasan

typedef struct nodeBalasan *AddressTree;
typedef struct nodeBalasan
{
    Balasan info;
    struct nodeBalasan *left;
    struct nodeBalasan *right;
} NodeBalasan;

typedef struct {
    AddressTree parentNode;
    int nEff;
} TreeBalasan;

#define INFOREP(t) (t)->info
#define LEFT(t) (t)->left
#define RIGHT(t) (t)->right

#define PARNODE(t) (t).parentNode

void createBalasan(int id, Word author, Word text, DATETIME time, int
idAuthorBalasan, Balasan *balasan);

#endif

```

```

#ifndef __TREEBALASAN_H__
#define __TREEBALASAN_H__

```

```

#include <stdio.h>
#include <stdlib.h>
#include "reply.h"
#include "../lib/boolean.h"

/* Constructor */
AddressTree CreateNewNode(Balasan balasan);

void CreateTreeBalasan(TreeBalasan *t);

/* Boolean Operation */

boolean isAddressTreeEmpty(AddressTree t);

boolean isOneElementTree(AddressTree t);

boolean doesNodeHaveBalasan(AddressTree t);

/* Search Operation */

AddressTree searchBalasan(AddressTree t, int idFind);

/* Add Operation */

void addBalasan(AddressTree *n, Balasan balasan);

/* Delete Operation */

void deleteAllNode(AddressTree n);

int deleteTreeBalasan(AddressTree *t, int idFind, int currID);

/* Display */
void displayBalasanPublic(Balasan bal, int dep);

```



```
void displayBalasanPrivate(Balasan bal, int dep);
void skipTab(int dep);

#endif
```

### 3.10 ADT Graf dengan Representasi Adjacency Matrix

Penjelasan sketsa struktur data: terdapat struktur data Graph yang terdiri atas adjMat yang merupakan matriks ketetanggaan dengan ukuran 20 \* 20 serta Neff bertipe integer.

Prototipe dan primitif pada ADT ini antara lain:

- konstruktor untuk membentuk graf dan graf pertemanan;
- predikat untuk mengecek teman dan permintaan pertemanan;
- selektor untuk mengatur serta membatalkan pertemanan dan permintaan pertemanan; serta
- operasi untuk menghitung jumlah teman.

Persoalan yang diselesaikan: modifikasi pertemanan.

Alasan pemilihan: karena hubungan pertemanan antarapengguna paling mudah dan efisien diimplementasikan dalam bentuk matriks.

Diimplementasikan sebagai ADT graf dengan representasi adjacency matriks dengan file header “graph.h”

Berikut header dari ADT graf dan friend.

```
#ifndef GRAPH_H
#define GRAPH_H
#include "../boolean.h"

typedef struct
{
    boolean adjMat[20][20];
    int Neff;
} Graph;

void CreateGraph(Graph *graph);
```

```
#endif
```

```
#ifndef FRIEND_H
#define FRIEND_H
#include <stdio.h>
#include "../..../lib/charmachine/charmachine.h"
#include "../..../lib/wordmachine/wordmachine.h"
#include "../..../lib/graph/graph.h"
#define Friend Graph // menggunakan ADT Graph dengan representasi
adjacency matrix

// CONSTRUCTOR
void CreateFriend(Friend *friend);

// CHECK
boolean isFriend(Friend friend, int id1, int id2);

boolean isRequested(Friend friend, int id1, int id2);

// SELECTOR
void setFriend(Friend *friend, int id1, int id2);

void unsetFriend(Friend *friend, int id1, int id2);

void setRequest(Friend *friend, int id1, int id2);

void unsetRequest(Friend *friend, int id1, int id2);

/*OPERATION*/
int countFriend(Friend friend, int currID);

#endif
```



## 4 Program Utama

Pertama-tama, program utama dengan nama file “main.c” akan meng-include seluruh file header yang ada dalam folder Tubes01-IF2110. Kemudian pengguna dapat memasukkan beberapa *command* yang sesuai dengan daftar *command* yang telah ada pada program. Jika *command* yang dimasukkan tidak sesuai, maka program akan meminta pengguna untuk memasukkan *command* kembali. Pengguna tidak akan keluar dari program dan data-data yang telah ada tidak akan terhapus. Program menggunakan mesin karakter dan mesin kata untuk membaca input dari pengguna.

Terdapat beberapa *command* yang dapat digunakan oleh pengguna, antara lain DAFTAR, MASUK, KELUAR, TUTUP\_PROGRAM, GANTI\_PROFIL, LIHAT\_PROFIL, ATUR\_JENIS\_AKUN, UBAH\_FOTO\_PROFIL, DAFTAR\_TEMAN, HAPUS\_TEMAN, TAMBAH\_TEMAN, DAFTAR\_PERMINTAAN\_PERTEMANAN, SETUJUI\_PERTEMANAN, KICAU, KICAUAN, SUKA\_KICAUAN [IDKicau], UBAH\_KICAUAN [IDKicau], BALAS [IDKicau] [IDBalasan], BALASAN [IDKicau], HAPUS\_BALASAN [IDKicau] [IDBalasan], BUAT\_DRAF, LIHAT\_DRAF, UTAS [IDKicau], SAMBUNG\_UTAS [IDUtas] [index], HAPUS\_UTAS [IDUtas] [index], CETAK\_UTAS [IDUtas], SIMPAN, serta MUAT.

Jika pengguna memasukkan *command* DAFTAR, program akan meminta pengguna untuk memasukkan nama dan kata sandi. Jika nama dan kata sandi valid maka pengguna akan dimasukkan ke dalam list pengguna. Jika pengguna memasukkan *command* MASUK, program akan meminta pengguna untuk memasukkan nama dan kata sandi dan jika keduanya sesuai dengan data dalam list pengguna, maka pengguna dapat masuk. Selain itu, pengguna dapat keluar dengan memasukkan *command* KELUAR hanya jika pengguna telah masuk sebelumnya. Jika pengguna memasukkan *command* TUTUP\_PROGRAM, maka seluruh data yang telah disimpan akan terhapus kecuali pengguna telah melakukan SIMPAN terlebih dahulu.

Jika pengguna memasukkan *command* GANTI\_PROFIL, maka program akan mengakses dan mengubah elemen dari list pengguna. Jika pengguna memasukkan LIHAT\_PROFIL, maka program akan menampilkan data dari elemen list pengguna yang dituju. Jika pengguna memasukkan *command* ATUR\_JENIS\_AKUN, maka program akan mengubah accType dari elemen list pengguna yang sedang melakukan penggantian jenis akun. Jenis akun

diimplementasikan dalam bentuk boolean, dimana true (1) menandakan akun tersebut *public*, sedangkan false (0) menandakan akun tersebut *private*. Jika pengguna memasukkan *command* UBAH\_FOTO\_PROFIL, maka program akan mengakses matriks foto profil dari elemen list pengguna yang dituju (pengguna yang sedang login) kemudian mengubahnya.

Jika pengguna memasukkan *command* DAFTAR\_TEMAN, maka program akan mengakses graf pertemanan yang diimplementasikan dalam bentuk adjacency matriks kemudian menghitung ada berapa yang bernilai true (1) kemudian menampilkan pengguna-pengguna yang hubungan dalam matriksnya true dengan pengguna yang sedang login. Jika pengguna memasukkan *command* HAPUS\_TEMAN, pertama-tama program akan memvalidasinya terlebih dahulu, jika pengguna memasukkan *command* YA, maka program akan mengubah graf pertemanan dari pengguna yang sedang login dengan pengguna yang dituju dari 1 menjadi nol. Pengguna hanya dapat menghapus teman yang berteman dengannya.

Jika pengguna memasukkan *command* TAMBAH\_TEMAN maka program akan meminta pengguna untuk memasukkan nama pengguna yang ingin dijadikan teman, kemudian pengguna tadi akan dimasukkan (enqueue) dalam priority queue permintaan pertemanan dari pengguna yang ingin diajak berteman sesuai dengan prioritas berdasarkan jumlah teman dari pengguna tersebut. Jika pengguna memasukkan *command* DAFTAR\_PERMINTAAN\_PERTEMANAN, maka program akan mengakses priority queue permintaan pertemanan dari pengguna tersebut dan menampilkan nama-nama pengguna dan jumlah teman dari pengguna-pengguna tersebut. Jika pengguna memasukkan *command* SETUJUI\_PERTEMANAN, maka program akan mengakses priority queue permintaan pertemanan dari pengguna tersebut. Jika permintaan pertemanan disetujui maka program akan melakukan dequeue dari priority queue tersebut, kemudian graf pertemanan kedua pengguna tersebut akan diubah dari 0 (false) menjadi 1 (true).

Jika pengguna memasukkan *command* KICAU maka kicauan akan ditambahkan ke dalam list kicauan setelah divalidasi bahwa isi dari kicau tersebut tidak hanya berisi spasi. Jika kicauan yang dimasukkan lebih dari 280 karakter maka karakter yang diambil hanya 280 karakter pertama. Terdapat global variabel untuk menghitung id kicauan terakhir dan tiap pengguna berhasil membuat kicauan baru, maka variabel tersebut akan bertambah satu. Jika pengguna memasukkan *command* KICAUAN, maka program akan menampilkan seluruh kicauan dari dirinya sendiri dengan urutan dari yang terbaru (ID paling besar). Info yang

ditampilkan antara lain ID, author, waktu dan tanggal kicau, isi kicauan, jumlah *like* beserta hashtag. Jika pengguna memasukkan command SUKA\_KICAUAN [IDKicau], maka program akan mengakses elemen *like* dari *struct* kicauan dari list kicauan dengan id yang dituju kemudian menambahkannya dengan satu. Jika pengguna memasukkan *command* UBAH\_KICAUAN [IDKicau], maka program akan mengakses elemen buffer dari list kicauan kemudian mengubah isinya sesuai dengan input dari pengguna.

Jika pengguna memasukkan *command* BALAS [IDKicau] [IDBalasan], maka pengguna dapat membalas balasan dengan indeks [IDBalasan] yang terdapat pada kicauan dengan indeks [IDKicau]. Jika balasan berhasil dibuat, maka akan terbentuk *node* balasan baru yang akan ditambahkan pada left dari node balasan dengan id [IDBalasan] jika *left* dari *node* tersebut sama dengan null, atau program akan melakukan *traverse* ke kanan dari *left node* yang di balas dan *node* balasan yang baru akan ditambahkan pada ujung paling kanan. Jika IDBalasan yang dimasukkan -1 artinya program akan membentuk root tree baru dari kicauan dengan indeks IDKicau. Jika pengguna memasukkan *command* BALASAN [IDKicau] maka program akan menampilkan seluruh balasan dari kicauan dengan indeks [IDKicau] dari balasan yang menjadi root dari tree balasan kicauan tersebut kemudian di-*traverse* ke kiri lalu ke kanan. Indentasi tiap balasan ditampilkan sesuai dengan level node balasan dalam treenya, dimana dalam program ini, left dari suatu node akan memiliki level yang lebih tinggi namun *right* dari suatu *node* akan memiliki level yang sama dengan *node* tersebut. Informasi yang ditampilkan antara lain ID, penulis, waktu perilis balasan (jam, menit, detik, tanggal, bulan, serta tahun), dan text balasan. Jika pengguna memasukkan command HAPUS\_BALASAN [IDKicau] [IDBalasan] maka program akan melakukan pencarian *node* balasan dengan id IDBalasan kemudian menghapus seluruh *node* dari *left node* balasan yang dimaksud secara rekursif. Kemudian *pointer* yang menunjuk pada dirinya akan dipindahkan ke *node* balasan selanjutnya (*right* dari *node* tersebut jika ada). Selanjutnya *node* yang sudah diputus *pointer*-nya tadi di dealokasi.

Jika pengguna memasukkan *command* BUAT\_DRAF maka pengguna akan diminta untuk memasukkan teks yang merupakan konten dari draft. Setelah itu pengguna dapat melakukan SIMPAN, HAPUS, maupun TERBIT. Jika pengguna memasukkan command SIMPAN, maka draf tersebut akan dimasukkan ke dalam stack draf pengguna tersebut sebagai elemen puncak (*top*) dari stack tersebut. Jika pengguna memasukkan *command* HAPUS, maka draf yang telah

dibuat akan langsung dihapus tanpa pernah di-*push* ke dalam stack. Jika pengguna memasukkan *command* TERBIT, maka program akan menambahkan draf tersebut ke dalam list kicauan. Jika pengguna memasukkan *command* LIHAT\_DRAF maka program akan mengakses elemen puncak (*top*) dari stack draf pengguna tersebut kemudian ditampilkan. Kemudian pengguna dapat memasukkan *command* HAPUS, UBAH, dan TERBIT. mekanisme HAPUS dan TERBIT sama dengan penjelasan di atas. Jika pengguna memasukkan *command* UBAH, maka program akan melakukan *pop* stack draf pengguna tersebut, mengedit elemen-elemennya sesuai dengan masukan dari pengguna, kemudian melakukan *push* ke dalam stack tersebut. Pengguna hanya dapat melihat draf terakhir yang disimpan karena stack hanya dapat mengakses elemen puncak dari dirinya sendiri.

Jika pengguna memasukkan *command* SIMPAN, maka program akan meminta pengguna untuk memasukkan nama folder yang akan digunakan untuk menyimpan data, kemudian program akan membuat file config dari pengguna, kicauan, balasan, utas, dan draf sehingga data-data tersebut dapat digunakan lagi setelah pengguna memasukkan *command* TUTUP\_PROGRAM. Jika pengguna memasukkan *command* MUAT, maka program akan meminta pengguna memasukkan nama folder yang akan dimuat. Jika folder tersebut ada, maka program akan mengakses file-file config dari folder tersebut agar dapat dilanjutkan.

## 5 Algoritma-Algoritma Menarik

### 5.1 Penghapusan node tree secara cascade.

Algoritma tersebut digunakan pada ADT tree untuk menghapus sebuah node beserta seluruh childnya. Algoritma ini menarik karena dapat menghapus node secara efisien.



## 6 Data Test

### 6.1 INISIALISASI

### 6.2 DAFTAR

*Command* DAFTAR digunakan untuk mendaftarkan akun baru pada aplikasi BurBir. Pada awalnya program akan meminta nama user dan password.

```
>> DAFTAR;
Masukkan nama:
adril;

Masukkan password:
123;

Anda telah berhasil masuk dengan nama pengguna adril. Mari menjelajahi BurBir bersama Ande-Ande Lumut!
```

Gambar 6.2.1 Tampilan ketika user ingin mendaftar akun baru

```
>> DAFTAR;
Anda sudah masuk. Keluar terlebih dahulu untuk melakukan daftar.
```

Gambar 6.2.2 Tampilan ketika user ingin mendaftar tapi sudah masuk

### 6.3 MASUK

*Command* MASUK digunakan untuk masuk ke akun yang sudah dibuat sebelumnya. Pada awalnya program akan meminta nama serta password akun yang ingin dimasuki

```
>> MASUK;
Masukkan nama:
adril;

Masukkan password:
123;

Anda telah berhasil masuk dengan nama pengguna adril. Mari menjelajahi BurBir bersama Ande-Ande Lumut!
```

Gambar 6.3.1. Tampilan ketika user ingin memasuki akun BurBir

```
>> MASUK;  
Wah Anda sudah masuk. Keluar dulu yuk!
```

Gambar 6.3.2 Tampilan ketika user ingin masuk tetapi user sudah masuk ke akun sebelumnya

## 6.4 KELUAR

*Command* KELUAR digunakan untuk keluar dari akun. Jika pada awalnya user tidak login, maka program akan meminta user untuk login terlebih dahulu.

```
>> KELUAR;  
Keluar dari akun...
```

Gambar 6.4.1 Tampilan ketika user berhasil keluar dari akun

```
>> KELUAR;  
Anda memang tidak pernah masuk...
```

Gambar 6.4.2 Tampilan ketika user tidak berhasil keluar dari akun

## 6.5 TUTUP\_PROGRAM

Ketika selesai menggunakan aplikasi BurBir, maka pengguna memiliki opsi untuk menutup program. *Command* ini dapat dilakukan saat aplikasi meminta *command*, yakni ketika tanda (>>) muncul

```
>> TUTUP_PROGRAM;  
Selesai
```

Gambar 6.5.1 Tampilan ketika user menutup program

## 6.6 GANTI\_PROFIL

*Command* GANTI\_PROFIL digunakan untuk mengganti profil pengguna. Program akan meminta Bio akun, nomor telepon, serta weton pengguna.

```
>> GANTI_PROFIL;
| Nama: adril
| Bio Akun:
| No HP:
| Weton:

Foto profil akun adril
*****
*****
*****
*****
*****

Masukkan Bio Akun:
saya adril;

Masukkan No HP:
085888;

Masukkan Weton:
Pon;

Profil Anda sudah berhasil diperbaharui
```

Gambar 6.6.1 Tamplan ketika user mengganti profil

## 6.7 LIHAT\_PROFIL

*Command* LIHAT\_PROFIL digunakan untuk melihat profil user. *Command* ini dibatasi oleh jenis akun yang ingin dilihat profilnya. Untuk akun publik, user dapat melihat tanpa berteman dengan. Untuk akun privat, user perlu berteman dengan akun tersebut terlebih dahulu.

```
>> LIHAT_PROFIL marvel;
| Nama: marvel
| Bio Akun:
| No HP:
| Weton:

Foto profil akun marvel
*****
*****
*****
*****
*****

>> |
```

Gambar 6.7.1 Tampilan ketika user melihat akun publik

```
>> LIHAT_PROFIL marvel;
Wah, akun marvel diprivat nih. Ikuti dulu yuk untuk bisa melihat profil marvel
```

Gambar 6.7.2 Tampilan ketika user melihat akun privat

## 6.8 ATUR\_JENIS\_AKUN

*Command* ATUR\_JENIS\_AKUN digunakan untuk mengubah jenis akun user. Secara *default*, akun user bersifat publik dan dapat dilihat oleh akun lain. Dengan menggunakan *command* ini, user dapat mengubah jenis akun dari publik ke privat dan sebaliknya.

```
>> ATUR_JENIS_AKUN;
Saat ini, akun Anda adalah akun Publik.
Ingin mengubah ke Akun Privat?
(YA/TIDAK) YA;
```

Gambar 6.8.1 Tampilan ketika user menggantikan jenis akun

## 6.9 UBAH\_FOTO\_PROFIL

*Command* UBAH\_FOTO\_PROFIL digunakan untuk mengubah foto profil user dengan memasukkan simbol serta warna dari foto profil.

```
>> UBAH_FOTO_PROFIL;
Foto profil Anda saat ini adalah
*****
*****
*****
*****
*****

Masukkan foto profil yang baru
R * R * R * R * R *
R * G @ B * G @ R *
R * G @ G @ G @ R *
R * G @ B * G @ R *
R * R * R * R * R *
;
>> LIHAT_PROFIL adrill;
| Nama: adrill
| Bio Akun:
| No HP:
| Weton:

Foto profil akun adrill
*****
* @ * @ *
* @ @ @ *
* @ * @ *
*****
```

Gambar 6.9.1 Tampilan ketika user menggantikan foto profil serta hasil perubahan.

## 6.10 DAFTAR\_TEMAN

*Command* DAFTAR\_TEMAN merupakan *command* untuk menampilkan daftar teman user.

```
>> DAFTAR_TEMAN;
adril memiliki 2 teman
Daftar teman adril
| marvel
| eve
```

Gambar 6.10.1 Tampilan ketika user menampilkan daftar temannya

## 6.11 HAPUS\_TEMAN

Command HAPUS\_TEMAN digunakan untuk menghapus teman dari daftar teman user.

```
>> DAFTAR_TEMAN;
eve memiliki 1 teman
Daftar teman eve
| adril

>> HAPUS_TEMAN;
Masukkan nama pengguna:
adril;

Apakah anda yakin ingin menghapus adril dari daftar teman anda? (YA/TIDAK) YA;
adril berhasil dihapus dari teman Anda.

>> DAFTAR_TEMAN;
eve belum mempunyai teman
```

Gambar 6.11.1 Tampilan ketika user hapus teman dan hasilnya

## 6.12 TAMBAH\_TEMAN

Command TAMBAH\_TEMAN digunakan untuk memberi permintaan pertemanan kepada user lain. Program akan meminta nama user yang ingin diberikan permintaan pertemanan.

```
>> TAMBAH_TEMAN;  
Masukkan nama pengguna:  
adril;  
  
Permintaan pertemanan kepada adril telah dikirim. Tunggu beberapa saat hingga permintaan Anda disetujui.
```

Gambar 6.12.1 Tampilan ketika user mengirimkan permintaan pertemanan

## 6.13 DAFTAR\_PERMINTAAN\_PERTEMANAN

*Command* ini akan menampilkan permintaan pertemanan kepada user, permintaan pertemanan akan diurutkan berdasarkan kepopuleran pengirim pada saat itu.

```
>> DAFTAR_PERMINTAAN_PERTEMANAN;  
Terdapat 5 permintaan pertemanan untuk Anda  
  
| eve  
| Jumlah teman: 4  
  
| saad  
| Jumlah teman: 3  
  
| marvel  
| Jumlah teman: 3  
  
| 123  
| Jumlah teman: 1  
  
| Dio Brando  
| Jumlah teman: 0  
  
>> □
```



Gambar 6.13.1 Tampilan ketika user meminta daftar pertemanan

## 6.14 SETUJUI\_PERTEMANAN

*Command* SETUJUI\_PERTEMANAN digunakan untuk menerima/menolak pertemanan, pengguna hanya dapat menyetujui permintaan pertemanan dari pengirim yang memiliki teman paling banyak sampai yang paling sedikit.

```
>> SETUJUI_PERTEMANAN;
Permintaan pertemanan teratas dari eve

| eve
| Jumlah teman: 4

Apakah Anda ingin menyetujui permintaan pertemanan ini? (YA/TIDAK)  YA;

Permintaan pertemanan dari eve telah disetujui. Selamat! Anda telah berteman dengan eve.
```

Gambar 6.14.1 Tampilan ketika user menyetujui pertemanan.

```
>> SETUJUI_PERTEMANAN;
Permintaan pertemanan teratas dari saad

| saad
| Jumlah teman: 3

Apakah Anda ingin menyetujui permintaan pertemanan ini? (YA/TIDAK)  TIDAK;

Permintaan pertemanan dari saad telah ditolak
```

Gambar 6.14.2 Tampilan ketika user menolak pertemanan

## 6.15 KICAU

*Command* KICAU digunakan untuk membuat sebuah kicau. Program akan meminta isi dari kicau dan hashtag, lalu program akan menampilkan kicau tersebut.

```
>> KICAU;
Masukkan kicauan:
4 jam lagi deadline tubes alstrukdat;

Masukkan hashtag:
panik_mode;

Selamat! kicauan telah diterbitkan!
Detil kicauan:
| ID = 1
| myAcc
| 24/11/2023 17:04:08
| 4 jam lagi deadline tubes alstrukdat
| #panik_mode
| Disukai: 0
```

Gambar 6.15.1 Tampilan ketika user membuat kicau.

## 6.16 KICAUAN

*Command* KICAUAN menampilkan kicau milik user serta teman-temannya terurut dari yang terbaru sampai terlama

```

>> KICAUAN;
| ID = 5
| adrill
| 24/11/2023 17:51:53
| ini punya adrill terbaru
| #new_hash
| Disukai: 0

| ID = 4
| marvel
| 24/11/2023 17:50:46
| kicau ketiga marvel
| #this_is_hard
| Disukai: 0

| ID = 3
| adrill
| 24/11/2023 17:50:03
| ini kicau 1 adrill
| #this_is_terrible
| Disukai: 0

| ID = 2
| marvel
| 24/11/2023 17:49:34
| ini kicau kedua marvel
| #what_is_life
| Disukai: 0

| ID = 1
| marvel
| 24/11/2023 17:49:05
| saya marvel
| #this_is_my_life
| Disukai: 0

```

Gambar 6.16.1 Tampilan ketika user menggunakan command KICAUAN

## 6.17 SUKA\_KICAUAN

*Command* SUKA\_KICAUAN digunakan untuk menyukai sebuah kicau. Program ini akan meminta id kicau yang ingin disukai. User dapat menyukai kicau diri sendiri serta kicau user lain yang publik atau teman user.

```
>> SUKA_KICAUAN 1;
Selamat! kicauan telah disukai!
Detil kicauan:
| ID = 1
| marvel
| 24/11/2023 17:49:05
| saya marvel
| #this_is_my_life
| Disukai: 1
```

Gambar 6.17.1 Tampilan ketika user berhasil menyukai kicau user lain

```
>> SUKA_KICAUAN 6;
Wah, kicauan tersebut dibuat oleh akun privat! Ikuti akun itu dulu
```

Gambar 6.17.2 Tampilan ketika user tidak berhasil menyukai kicau

## 6.18 UBAH\_KICAUAN

*Command* ini digunakan untuk mengubah kicau user. Program akan meminta id kicau yang ingin diubah, jika bukan kicau yang dimiliki user / id kicau tidak maka program akan menampilkan pesan gagal.

```
Selamat! kicauan telah diterbitkan!
| ID = 5
| adril
| 24/11/2023 17:51:53
| ini punya adril yang diubah
| #new_hash
| Disukai: 0
```

Gambar 6.18.1 Tampilan ketika user berhasil menggantikan kicau

```
>> UBAH_KICAUAN 1;  
Kicauan dengan ID = 1 bukan milikmu!
```

Gambar 6.18.2 Tampilan ketika user mencoba untuk mengubah kicau punya user lain

```
>> UBAH_KICAUAN -1;  
Tidak ditemukan kicauan dengan ID = -1!
```

Gambar 6.18.3 Tampilan ketika user mencoba untuk mengubah kicau yang tidak ada (tidak ada kicau dengan id tsb)

## 6.19 BALAS

*Command* BALAS digunakan untuk membalas kicau. Program akan meminta IDKicau dan IDBalasan. setiap balasan pada kicau memiliki id masing-masing, hal tersebut merupakan (IDBalasan), untuk membalas kicau maka menggunakan -1 sebagai IDBalasan.

## 6.20 BALASAN

*Command* BALASAN digunakan untuk menampilkan balasan dari sebuah id kicau. Program akan meminta idKicau yang ingin dilihat balasannya, tetapi

```

>> BALASAN 5;
| ID = 1
| adril
| 24/11/2023 18:35:25
| balasan pertama

| ID = 2
| adril
| 24/11/2023 18:38:47
| balasan dari balasan

| ID = 3
| adril
| 24/11/2023 18:38:57
| balasan kicau baru

| ID = 4
| PRIVAT
| PRIVAT
| PRIVAT

```

Gambar 6.20.1 Tampilan ketika pengguna mencoba untuk melihat balasan pada kicau dengan id

5

```

>> BALASAN -1;
Wah, tidak terdapat kicauan yang dapat ditampilkan!

```

Gambar 6.20.2 Tampilan ketika pengguna ingin menampilkan balasan dari kicau yang tidak ada

Pada gambar 6.20.1, pengguna tidak berteman dengan pembalas yang id balasan 4, sehingga tampilan akan privat

## 6.21 HAPUS\_BALASAN

*Command* HAPUS\_BALASAN digunakan untuk menghapus sebuah balasan. Program akan meminta idKicau dan idBalasan, dengan kedua id tersebut, program akan menghapus balasan yang diinginkan pengguna. Balasan yang dapat dihapus merupakan balasan milik pengguna sendiri.

```
>> HAPUS_BALASAN 5 1;
Balasan berhasil dihapus

>> BALASAN 5;
| ID = 3
| adril
| 24/11/2023 18:38:57
| balasan kicau baru

| ID = 4
| PRIVAT
| PRIVAT
| PRIVAT
```

Gambar 6.21.1 Tampilan ketika balasan berhasil dihapus serta hasilnya

```
>> HAPUS_BALASAN -1 -1;
Wah, tidak terdapat kicauan yang ingin Anda hapus!
```

Gambar 6.21.2 Tampilan ketika balasan tidak berhasil dihapus (kicau dengan input id tidak ada)

```
>> HAPUS_BALASAN 5 4;
Hei, ini balasan punya siapa? Jangan dihapus ya!
```

Gambar 6.21.3 Tampilan ketika balasan yang ingin dihapus bukan milik user

## 6.22 BUAT\_DRAF

*Command* ini digunakan untuk membuat draf,

```

>> BUAT_DRAF;
Masukkan draf:
halo semua 2;

Masukkan hashtag:
this_is_hard;
Apakah anda ingin menghapus, menyimpan, atau menerbitkan draf ini?
HAPUS;

Draf telah berhasil dihapus!

```

Gambar 6.22.1 Tampilan ketika user ingin menghapus draf yang dibuat

```

>> BUAT_DRAF;
Masukkan draf:
draf baru;

Masukkan hashtag:
life;
Apakah anda ingin menghapus, menyimpan, atau menerbitkan draf ini?
SIMPAN;

Draf telah berhasil disimpan!

```

Gambar 6.22.2 Tampilkan ketika user menyimpan draf yang sudah dibuat

```

>> BUAT_DRAF;
Masukkan draf:
terbit sekarang;

Masukkan hashtag:
terbit1;
Apakah anda ingin menghapus, menyimpan, atau menerbitkan draf ini?
TERBIT;

Selamat! Draf kicauan telah diterbitkan!
Detil kicauan:
| ID = 8
| adril
| 24/11/2023 19:15:44
| terbit sekarang
| #terbit1
| Disukai: 0

```

Gambar 6.22.3 Tampilan ketika user ingin langsung menerbitkan draf yang dibuat



## 6.23 LIHAT\_DRAF

*Command* LIHAT\_DRAF digunakan untuk melihat draf yang sudah dibuat pengguna

```
>> LIHAT_DRAF;  
  
Yah, anda belum memiliki draf apapun! Buat dulu ya :D
```

Gambar 6.23.1 Tampilan ketika pengguna ingin melihat draf tetapi pengguna belum membuat draf sebelumnya

```
>> LIHAT_DRAF;  
  
Ini draf terakhir Anda:  
| 24/11/2023 19:12:29  
| draf baru  
  
Apakah anda ingin mengubah, menghapus, atau menerbitkan draf ini? (KEMBALI jika ingin kembali)  
TERBIT;  
  
Selamat! Draft kicauan telah diterbitkan!  
Detail kicauan:  
| ID = 9  
| adrill  
| 24/11/2023 19:40:51  
| draf baru  
| #life  
| Disukai: 0
```

Gambar 6.23.2 Tampilkan ketika pengguna ingin melihat draf, lalu menerbitkan draf tersebut

```
>> LIHAT_DRAF;  
  
Ini draf terakhir Anda:  
| 24/11/2023 19:41:56  
| draf lagi  
  
Apakah anda ingin mengubah, menghapus, atau menerbitkan draf ini? (KEMBALI jika ingin kembali)  
HAPUS;  
  
Draf telah berhasil dihapus!
```

Gambar 6.23.3 Tampilan ketika pengguna ingin melihat draf, lalu menghapus draf tersebut

```
Draf telah berhasil disimpan!
>> LIHAT_DRAF;

Ini draf terakhir Anda:
| 24/11/2023 19:44:17
| draf again

Apakah anda ingin mengubah, menghapus, atau menerbitkan draf ini? (KEMBALI jika ingin kembali)
UBAH;

Masukkan draf yang baru:
draf ubah again;

Masukkan hashtag:
ubah_hash;
Apakah anda ingin menghapus, menyimpan, atau menerbitkan draf ini?
SIMPAN;

Draf telah berhasil disimpan!
>> LIHAT_DRAF;

Ini draf terakhir Anda:
| 24/11/2023 19:44:48
| draf ubah again

Apakah anda ingin mengubah, menghapus, atau menerbitkan draf ini? (KEMBALI jika ingin kembali)
█
```

Gambar 6.23.4 Tampilan ketika user ingin melihat draf, lalu mengubah draf tersebut

## 6.24 UTAS

*Command* UTAS digunakan untuk membuat utas pada sebuah kicau (kicau tersebut menjadi kicauan utama). Program akan meminta IdKicau yang ingin dibuat menjadi kicau utama.

```

>> UTAS 3;

Utas berhasil dibuat!

Masukkan kicauan:
utas 1 adrıl;

Apakah Anda ingin melanjutkan utas ini?
YES;

Masukkan kicauan:
utas 2 adrıl;

Apakah Anda ingin melanjutkan utas ini?
YES;

Masukkan kicauan:
utas 3 adrıl;

Apakah Anda ingin melanjutkan utas ini?
TIDAK;
Utas selesai!

```

Gambar 6.24.1 Tampilan ketika pengguna berhasil membuat utas pada kicau dengan id 3

```

>> UTAS 2;
Utas ini bukan milik anda!

```

Gambar 6.24.2 Tampilan ketika pengguna mencoba untuk membuat utas pada kicau yang bukan milik pengguna

```

>> UTAS -1;
Kicauan tidak ditemukan

```

Gambar 6.24.3 Tampilan ketika penggunaan mencoba membuat utas pada kicau yang tidak ada (kicau dengan id tersebut tidak ada)

```

>> UTAS 3;
Kicauan ini sudah merupakan utas!

```

Gambar 6.24.4 Tampilan ketika pengguna mencoba untuk membuat utas pada kicau yang sudah memiliki utas / merupakan kicauan utama

## 6.25 SAMBUNG\_UTAS

*Command* SAMBUNG\_UTAS digunakan untuk menyambung sebuah utas yang sudah ada (menambahkan). Program menerima idUtas dan index yang ingin disambungkan.

```
>> SAMBUNG_UTAS 1 1;
Masukkan kicauan:
halo;
Utas berhasil disambungkan

>> CETAK_UTAS 1;
| ID = 3
| adr1l
| 24/11/2023 17:50:03
| ini kicau 1 adr1l

| INDEX = 1
| adr1l
| 24/11/2023 20:02:58
| halo

| INDEX = 2
| adr1l
| 24/11/2023 19:49:15
| utas 1 adr1l

| INDEX = 3
| adr1l
| 24/11/2023 19:49:44
| utas 2 adr1l

| INDEX = 4
| adr1l
| 24/11/2023 19:49:50
```

Gambar 6.25.1 Tampilan ketika pengguna berhasil menyambungkan utas

```
>> SAMBUNG_UTAS -1 -1;
Utas tidak ditemukan!
```

Gambar 6.25.2 Tampilan ketika pengguna tidak berhasil menyambungkan utas sebab utas dengan id tersebut tidak ada.

```
>> SAMBUNG_UTAS 1 6;  
Index terlalu tinggi!
```

Gambar 6.25.3 Tampilan ketika pengguna tidak berhasil menyambungkan utas sebab index yang diinput terlalu tinggi (index di luar utas)

```
>> SAMBUNG_UTAS 1 -1;  
Index terlalu rendah
```

Gambar 6.25.4 Tampilan ketika pengguna tidak berhasil dalam menyambungkan utas sebab index yang diinput terlalu rendah

```
>> SAMBUNG_UTAS 2 1;  
Anda tidak bisa menyambung utas ini!
```

Gambar 6.25.5 Tampilan ketika pengguna tidak berhasil menyambungkan utas sebab utas tersebut milik pengguna lain

## 6.26 HAPUS\_UTAS

*Command* HAPUS\_UTAS digunakan untuk menghapus utas pada index tertentu. Program menerima idUtas dan index utas lalu menghapus utas pada kicau tersebut.

```
>> HAPUS_UTAS -1 -1;  
Utas tidak ditemukan!
```

Gambar 6.26.1 Tampilan ketika pengguna gagal menghapus utas sebab utas dengan id input pengguna tidak ada

```
>> HAPUS_UTAS 1 0;  
Anda tidak bisa menghapus kicauan utama!
```

Gambar 6.26.2 Tampilan ketika pengguna gagal menghapus utas sebab kicauan utama (index 0) tidak dapat dihapus

```
>> HAPUS_UTAS 1 1;  
Kicauan sambungan berhasil dihapus!  
  
>> CETAK_UTAS 1;  
| ID = 3  
| adril  
| 24/11/2023 17:50:03  
| ini kicau 1 adril  
  
| INDEX = 1  
| adril  
| 24/11/2023 19:49:15  
| utas 1 adril  
  
| INDEX = 2  
| adril  
| 24/11/2023 19:49:44  
| utas 2 adril  
  
| INDEX = 3  
| adril  
| 24/11/2023 19:49:50  
| utas 3 adril
```

Gambar 6.26.3 Tampilan ketika utas berhasil dihapus, perhatikan gambar (6.25.1), utas index 1 dengan text “halo” berhasil terhapus

```
>> HAPUS_UTAS 2 1;  
Anda tidak bisa menghapus kicauan dalam utas ini!
```

Gambar 6.26.4 Tampilan ketika pengguna gagal dalam menghapus utas sebab utas tersebut milik pengguna lain

## 6.27 CETAK\_UTAS

*Command* digunakan untuk mencetak utas. Program akan meminta idUtas yang ingin dicetak, kemudian utas akan dicetak jika sesuai dengan syarat-syarat tertentu.

```
>> CETAK_UTAS 1;
| ID = 3
| adr1l
| 24/11/2023 17:50:03
| ini kicau 1 adr1l

| INDEX = 1
| adr1l
| 24/11/2023 19:49:15
| utas 1 adr1l

| INDEX = 2
| adr1l
| 24/11/2023 19:49:44
| utas 2 adr1l

| INDEX = 3
| adr1l
| 24/11/2023 19:49:50
| utas 3 adr1l
```

Gambar 6.27.1 Tampilan ketika pengguna berhasil mencetak utas

```
>> CETAK_UTAS 3;
Akun yang membuat utas ini adalah akun privat! Ikuti dahulu akun ini untuk melihat utasnya!
```

Gambar 6.27.2 Tampilan ketika pengguna tidak berhasil mencetak utas sebab pemilik utas dengan id 3 merupakan akun privat dan pengguna tidak berteman dengan akun tersebut

## 6.28 SIMPAN

```
Masukkan nama folder penyimpanan
config-2;

config/config-2
Belum terdapat config/config-2
Akan dilakukan pembuatan Folder1 terlebih dahulu.

Mohon tunggu...
1...
2...
3...

Anda akan melakukan penyimpanan di config/config-2

Mohon tunggu...
1...
2...
3...

Penyimpanan telah berhasil dilakukan!
```

Gambar 6.28.1 Simpan

## 6.29 MUAT

```
Masukkan nama folder yang hendak dimuat.
config-1;

Anda akan melakukan pemuatan dari config/config-1

14/10/2023 11:09:18
14/10/2023 11:09:12
14/10/2023 11:09:12
14/10/2023 11:09:12
config/config-1/utas.config
Mohon tunggu...
1...
2...
3...

Pemuatan selesai!
```

Gambar 6.29.1 Muat



## 7 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	DAFTAR	Memeriksa apakah pengguna berhasil melakukan daftar.	Memasukkan command DAFTAR kemudian memasukkan nama dan password yang valid.	Data test 6.2	Pengguna dapat melakukan daftar.	Pengguna dapat melakukan daftar.
	MASUK	Memeriksa apakah pengguna yang telah melakukan daftar dapat masuk.	Memasukkan command MASUK kemudian memasukkan nama dan password yang valid.	Data test 6.3	Pengguna dapat masuk.	Pengguna dapat masuk.
2	KELUAR	Memeriksa apakah pengguna yang telah masuk dapat keluar.	Memasukkan command KELUAR kemudian memasukkan nama dan password yang valid.	Data test 6.4.1	Pengguna dapat keluar.	Pengguna dapat keluar.
3	KELUAR	Memeriksa apakah pengguna yang belum masuk dapat keluar.	Memasukkan command KELUAR kemudian memasukkan nama dan password yang tidak valid.	Data test 6.4.2	Pengguna tidak dapat keluar.	Pengguna tidak dapat keluar.
...	TUTUP_PROGRAM	Memeriksa apakah program dapat ditutup.	Memasukkan command TUTUP_PROGRAM	Data test 6.5.1	Program dapat ditutup.	Program dapat ditutup.
5	GANTI_PROFILE	Memeriksa apakah profil pengguna dapat diubah.	Memasukkan command GANTI_PROFIL kemudian memasukkan bio, nomor HP, dan weton yang baru.	Data test 6.6.1	Profil dapat diperbarui.	Profil dapat diperbarui.

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	LIHAT_PROFIL	Mengecek apakah profil dari pengguna dengan akun publik dapat dilihat oleh pengguna lain.	Memasukkan command LIHAT_PROFIL [NAMA]	Data test 6.7.1	Profil dapat dilihat.	Profil dapat dilihat.
	LIHAT_PROFIL	Mengecek apakah profil dari pengguna dengan akun privat dapat dilihat oleh pengguna lain.	Memasukkan command LIHAT_PROFIL [NAMA]	Data test 6.7.2	Profil tidak dapat dilihat.	Profil tidak dapat dilihat.
6	ATUR_JENIS_AKUN	Mengecek apakah jenis akun pengguna dapat diubah (privat menjadi publik dan sebaliknya).	Memasukkan command ATUR_JENIS_AKUN	Data test 6.8.1	Akun berhasil diubah.	Akun berhasil diubah.
	UBAH_FOTO_PROFIL	Mengecek apakah foto profil pengguna dapat diubah	Memasukkan command UBAH_FOTO_PROFIL kemudian memasukkan simbol serta warna foto profil	Data test 6.9.1	Foto profil dapat diubah dan foto profil yang baru ditampilkan.	Foto profil dapat diubah dan foto profil yang baru ditampilkan.
	DAFTAR_TEMAN	Mengecek apakah program dapat menampilkan daftar teman dari pengguna.	Memasukkan command DAFTAR_TEMAN	Data test 6.10.1	Daftar teman dapat ditampilkan.	Daftar teman dapat ditampilkan.
	HAPUS_TEMAN	Mengecek apakah pengguna	Memasukkan command HAPUS_TEMAN, nama	Data test 6.11.1	Teman berhasil	Teman berhasil

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		dapat menghapus teman dari daftar teman pengguna.	pengguna yang ingin dihapus, lalu melakukan validasi		dihapus dari daftar teman.	dihapus dari daftar teman.
	TAMBAH_TEMAN	Mengecek apakah pengguna dapat mengirim permintaan pertemanan kepada pengguna lain	Memasukkan command TAMBAH_TEMAN dan nama pengguna yang ingin dijadikan teman.	Data test 6.12.1	Permintaan berhasil dikirim dan pengguna tersebut masuk ke dalam queue permintaan pertemanan dari pengguna yang ingin dijadikan teman.	Permintaan berhasil dikirim dan pengguna tersebut masuk ke dalam queue permintaan pertemanan dari pengguna yang ingin dijadikan teman.
	DAFTAR_PERMINTAAN_PERTEMANAN	Mengecek apakah pengguna dapat melihat daftar pengguna lain yang mengirimkan permintaan pertemanan pada dirinya.	Memasukkan command DAFTAR_PERMINTAAN_PERTEMANAN	Data test 6.13.1	Daftar permintaan pertemanan dapat ditampilkan.	Daftar permintaan pertemanan dapat ditampilkan.
	SETUJUI_PERTEMANAN	Mengecek apakah pengguna dapat menyetujui permintaan	Memasukkan command SETUJUI_PERTEMANAN kemudian melakukan validasi.	Data test 6.14.1	Teman dapat ditambahkan.	Teman dapat ditambahkan.

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		pertemanan dari pengguna lain.				
	SETUJUI_PERTEMANAN	Mengecek apakah pengguna dapat menolak permintaan pertemanan dari pengguna lain.	Memasukkan command SETUJUI_PERTEMANAN kemudian melakukan validasi.	Data test 6.14.2	Permintaan pertemanan dihapus dari daftar permintaan pertemanan.	Permintaan pertemanan dihapus dari daftar permintaan pertemanan
	KICAU	Mengecek apakah pengguna dapat membuat kicauan baru.	Memasukkan command KICAU, isi kicauan, dan hashtag.	Data test 6.15.1	Kicauan berhasil diterbitkan.	Kicauan berhasil diterbitkan.
	KICAUAN	Mengecek apakah daftar kicauan yang telah dibuat dapat ditampilkan.	Memasukkan command KICAUAN.	Data test 6.16.1	Daftar kicauan berhasil ditampilkan.	Daftar kicauan berhasil ditampilkan.
	SUKA_KICAUAN	Mengecek apakah pengguna dapat menambahkan like pada kicauan.	Memasukkan command SUKA_KICAUAN [IDKicau]	Data test 6.17.1	Like pada kicauan yang dituju bertambah satu.	Like pada kicauan yang dituju bertambah satu.
	SUKA_KICAUAN	Mengecek apakah pengguna dapat menambahkan like pada kicauan akun privat yang tidak berteman dengannya.	Memasukkan command SUKA_KICAUAN [IDKicau]	Data test 6.17.2	Like tidak dapat ditambahkan.	Like tidak dapat ditambahkan.

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	UBAH_KICAUAN	Mengecek apakah pengguna dapat mengganti kicauan miliknya sendiri.	Memasukkan command UBAH_KICAUAN [IDKicau] dan isi kicauan yang baru.	Data test 6.18.1	Kicauan berhasil diubah.	Kicauan berhasil diubah.
	UBAH_KICAUAN	Mengecek apakah pengguna dapat mengganti kicauan milik pengguna lain.	Memasukkan command UBAH_KICAUAN [IDKicau]	Data test 6.18.2	Kicauan tidak dapat diubah.	Kicauan tidak dapat diubah.
	UBAH_KICAUAN	Mengecek apakah pengguna dapat mengganti kicauan dengan ID yang tidak valid.	Memasukkan command UBAH_KICAUAN [IDKicau]	Data test 6.18.3	Program menampilkan pesan bahwa kicauan tidak ditemukan.	Program menampilkan kalimat bahwa kicauan tidak ditemukan.
	BALAS					
	BALASAN	Mengecek apakah program dapat menampilkan balasan dari kicauan yang dituju.	Memasukkan command BALASAN [IDKicau]	Data test 6.20.1	Program menampilkan semua balasan dari kicauan yang dituju.	Program menampilkan semua balasan dari kicauan yang dituju.
	BALASAN	Mengecek apakah program dapat menampilkan balasan dari kicauan yang tidak ada.	Memasukkan command BALASAN [IDKicau]	Data test 6.20.2	Program menampilkan pesan bahwa balasan tidak ditemukan.	Program menampilkan kalimat bahwa balasan tidak ditemukan.

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	HAPUS_BALASAN	Mengecek apakah pengguna dapat menghapus balasan miliknya sendiri.	Memasukkan command HAPUS_BALASAN [IDKicau] [IDBalasan]	Data test 6.21.1	Balasan berhasil dihapus.	Balasan berhasil dihapus.
	HAPUS_BALASAN	Mengecek apakah pengguna dapat menghapus balasan yang tidak ada.	Memasukkan command HAPUS_BALASAN [IDKicau] [IDBalasan]	Data test 6.21.2	Program menampilkan pesan bahwa balasan tidak ditemukan.	Program menampilkan kalimat bahwa balasan tidak ditemukan.
	HAPUS_BALASAN	Mengecek apakah pengguna dapat menghapus balasan milik orang lain.	Memasukkan command HAPUS_BALASAN [IDKicau] [IDBalasan]	Data test 6.21.3	Balasan tidak dapat dihapus.	Balasan tidak dapat dihapus.
	BUAT_DRAF	Mengecek apakah pengguna dapat membuat draf kemudian langsung dihapus.	Memasukkan command BUAT_DRAF, isi draf, dan hastag, lalu memasukkan command HAPUS.	Data test 6.22.1	Draf berhasil dihapus.	Draf berhasil dihapus.
	BUAT_DRAF	Mengecek apakah pengguna dapat membuat draf kemudian disimpan.	Memasukkan command BUAT_DRAF, isi draf, dan hastag, lalu memasukkan command SIMPAN.	Data test 6.22.2	Draf berhasil disimpan.	Draf berhasil disimpan.
	BUAT_DRAF	Mengecek apakah pengguna dapat membuat draf kemudian langsung	Memasukkan command BUAT_DRAF, isi draf, dan hastag, lalu memasukkan command TERBIT.	Data test 6.22.3	Draf berhasil diterbitkan sebagai kicauan baru.	Draf berhasil diterbitkan sebagai kicauan baru.

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		diterbitkan sebagai kicauan baru.				
	LIHAT_DRAF	Mengecek apakah pengguna dapat melihat draf jika pengguna tersebut belum memiliki draf.	Memasukkan command LIHAT_DRAF	Data test 6.23.1	Program menampilkan pesan bahwa draf tidak ada.	Program menampilkan pesan bahwa draf tidak ada.
	LIHAT_DRAF	Mengecek apakah pengguna dapat melihat draf jika pengguna tersebut memiliki draf.	Memasukkan command LIHAT_DRAF	Data test 6.23.2	Program menampilkan draf yang terakhir yang ditambahkan.	Program menampilkan draf yang terakhir yang ditambahkan.
	LIHAT_DRAF	Mengecek apakah pengguna dapat menghapus draf miliknya.	Memasukkan command LIHAT_DRAF, lalu HAPUS.	Data test 6.23.3	Draf berhasil dihapus.	Draf berhasil dihapus.
	LIHAT_DRAF	Mengecek apakah pengguna dapat mengedit draf miliknya.	Memasukkan command LIHAT_DRAF, lalu UBAH..	Data test 6.23.4	Draf berhasil diubah.	Draf berhasil diubah.
	UTAS [IDKicau]	Mengecek apakah pengguna dapat membuat utas pada kicauan miliknya sendiri.	Memasukkan command UTAS [IDKicau], isi utas, dan melakukan validasi.	Data test 6.24.1	Utas berhasil dibuat.	Utas berhasil dibuat.
	UTAS [IDKicau]	Mengecek apakah pengguna	Memasukkan command UTAS [IDKicau]	Data test 6.24.2	Program menampilkan	Program menampilkan

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		dapat membuat utas pada kicauan milik orang lain.			pesan bahwa kicauan yang ingin ditambahkan dengan utas bukan miliknya.	pesan bahwa kicauan yang ingin ditambahkan dengan utas bukan miliknya.
	UTAS [IDKicau]	Mengecek apakah pengguna dapat membuat utas pada kicauan yang tidak ada.	Memasukkan command UTAS [IDKicau]	Data test 6.24.3	Program menampilkan pesan bahwa kicauan yang ingin ditambahkan dengan utas tidak ditemukan.	Program menampilkan pesan bahwa kicauan yang ingin ditambahkan dengan utas tidak ditemukan.
	UTAS [IDKicau]	Mengecek apakah pengguna dapat membuat utas pada kicauan yang sudah memiliki utas.	Memasukkan command UTAS [IDKicau]	Data test 6.24.4	Program menampilkan pesan bahwa kicauan yang ingin ditambahkan dengan utas sudah merupakan utas.	Program menampilkan pesan bahwa kicauan yang ingin ditambahkan dengan utas sudah merupakan utas. .
	SAMBUNG _UTAS [IDKicau] [IDUtas]	Mengecek apakah pengguna dapat menyambung utas dari kicauan miliknya sendiri.	Memasukkan command SAMBUNG_UTAS [IDKicau] [IDUtas] dan isi utas yang ingin disambungkan.	Data test 6.25.1	Utas berhasil ditambahkan.	Utas berhasil ditambahkan.



No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
	SAMBUNG_UTAS [IDKicau] [IDUtas]	Mengecek apakah pengguna dapat menyambung utas yang tidak ada.	Memasukkan command SAMBUNG_UTAS [IDKicau] [IDUtas].	Data test 6.25.2	Program menampilkan pesan bahwa utas tidak ditemukan.	Program menampilkan pesan bahwa utas tidak ditemukan.
	SAMBUNG_UTAS [IDKicau] [IDUtas]	Mengecek apakah pengguna dapat menyambung utas dengan indeks yang terlalu tinggi.	Memasukkan command SAMBUNG_UTAS [IDKicau] [IDUtas].	Data test 6.25.3	Program menampilkan pesan bahwa indeks terlalu tinggi.	Program menampilkan pesan bahwa indeks terlalu tinggi.
	SAMBUNG_UTAS [IDKicau] [IDUtas]	Mengecek apakah pengguna dapat menyambung utas dengan indeks yang terlalu rendah.	Memasukkan command SAMBUNG_UTAS [IDKicau] [IDUtas].	Data test 6.25.4	Program menampilkan pesan bahwa indeks terlalu rendah.	Program menampilkan pesan bahwa indeks terlalu rendah.
	HAPUS_UTAS [IDKicau] [IDUtas]	Mengecek apakah pengguna dapat menghapus utas yang tidak ada.	Memasukkan command HAPUS_UTAS [IDKicau] [IDUtas].	Data test 6.26.1	Program menampilkan pesan bahwa utas tidak ditemukan.	Program menampilkan pesan bahwa utas tidak ditemukan.
	HAPUS_UTAS [IDKicau] [IDUtas]	Mengecek apakah pengguna dapat menghapus utas yang merupakan kicauan utama.	Memasukkan command HAPUS_UTAS [IDKicau] [IDUtas].	Data test 6.26.2	Program menampilkan pesan bahwa pengguna tidak dapat menghapus	Program menampilkan pesan bahwa pengguna tidak dapat menghapus

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					kicauan utama.	kicauan utama.
	HAPUS_ UTAS [IDKicau] [IDUtas]	Mengecek apakah pengguna dapat menghapus utas miliknya.	Memasukkan command HAPUS_ UTAS [IDKicau] [IDUtas].	Data test 6.26.3	Utas berhasil dihapus.	Utas berhasil dihapus.
	HAPUS_ UTAS [IDKicau] [IDUtas]	Mengecek apakah pengguna dapat menghapus utas milik pengguna lain.	Memasukkan command HAPUS_ UTAS [IDKicau] [IDUtas].	Data test 6.26.4	Program menampilkan pesan bahwa utas tidak dapat dihapus.	Program menampilkan pesan bahwa utas tidak dapat dihapus.
	CETAK_ UTAS [IDUtas]	Mengecek apakah pengguna dapat mencetak utas.	Memasukkan command CETAK_ UTAS [IDUtas]	Data test 6.27.1	Utas berhasil ditampilkan.	Utas berhasil ditampilkan.
	CETAK_ UTAS [IDUtas]	Mengecek apakah pengguna dapat mencetak utas dari pengguna privat yang tidak berteman dengannya.	Memasukkan command CETAK_ UTAS [IDUtas]	Data test 6.27.2	Utas tidak dapat ditampilkan.	Utas tidak dapat ditampilkan..
	SIMPAN	Mengecek apakah program berhasil disimpan.	Memasukkan command SIMPAN.	Data test 6.28.1	Program berhasil disimpan.	Program berhasil disimpan.
	MUAT	Mengecek apakah program berhasil dimuat .	Memasukkan command MUAT.	Data test 6.29.1	Program berhasil dimuat.	Program berhasil dimuat.



## 8 Pembagian Kerja dalam Kelompok

NIM	Nama	Pembagian Tugas
13522068	Adril Putra Merin	Membuat program pengguna, profil, teman, permintaan teman, tagar, kelompok teman, dan FYB
13522075	Marvel Pangondian	Membuat program utas dan kicauan
13522083	Evelyn Yosiana	Membuat program balasan dan menulis laporan
13522092	Sa'ad Abdul Hakim	Membuat program save dan load untuk menerima dan menulis teks ke file config
13522105	Fabian Radenta Bangun	Membuat program draf dan menulis laporan

## 9 Lampiran

### 9.1 Deskripsi Tugas Besar 1







Klenting Kuning sedang sedih karena dia dirundung oleh ibu tirinya di aplikasi sosial media yang sedang beken saat zaman itu, yaitu Y. Dia dirundung karena dia berhasil memikat hati Ande-Ande Lumut. Ibu tirinya kesal, karena menurutnya, yang pantas untuk menjadi pasangan dari Ande-Ande Lumut adalah saudara tiri dari Klenting Kuning, yaitu Klenting Biru dan Klenting Merah. Klenting Kuning dikumpulkan di Y Spaces, tempat live audio, yang kelak disebut sebagai Kuning Space untuk dirundung oleh ibu tirinya bersama dengan anak-anaknya.

“Kuning, saya ini perwakilan Klenting Biru dan Klenting Merah. Saya sudah panggil advokat saya untuk bawa kasus ini ke meja hijau. Jadi kamu jangan macam-macam ya!”.







Klenting Kuning yang mendengar ancaman dari Ibu tirinya pun ketakutan dengan ancaman tersebut. Apalagi, ibu tirinya ini merupakan seorang aktivis HAM yang ayahnya merupakan salah satu pejabat terkenal di zamannya, dan begitu pula adiknya. Kakak dari suaminya juga merupakan salah satu kepala pejabat yang terkenal. Yang paling menyeramkan, dosen dari ibu tirinya merupakan salah satu petinggi partai di zaman itu.

Ande-Ande Lumut yang iba mendengar kisah Klenting Kuning kemudian berencana untuk membuat pengganti sosial media yang beken di zaman itu dimana ibu tiri dan saudara jahatnya dicekal dari pendaftaran, yang kelak dinamai BurBir (Burung Biru). Akan tetapi, dia tidak pandai dalam pemrograman bahasa C, lebih-lebih dia acap kali menggunakan **Github Copilot** dan **ChatGPT** saat mengerjakan pra-praktikum IF2110 Algoritma & Struktur Data. Kalian, anak buah dari Yuyu Kangkang, diminta untuk membantu Ande-Ande Lumut. Yuk bantu Ande-Ande Lumut membuat aplikasi BurBir di CLI!

## 9.2 Notulen Rapat

Asistensi I	
Tanggal : 2 November 2023	Catatan Asistensi:
Tempat : Lab Pemrograman	
Kehadiran Anggota Kelompok:	
No	
NIM	
Tanda tangan	
1	
13522068	
	
2	
13522075	
	
3	
13522083	
	
4	
13522092	
	
5	
13522105	
	
6	
	Tanda Tangan Asisten:
	

Asistensi II

Tanggal : 17 November 2023	Catatan Asistensi:
Tempat : Lab Pemrograman	
Kehadiran Anggota Kelompok:	
No	
NIM	
Tanda tangan	
1 13522068 	
2 13522075 	
3 13522033 	
4 13522092 	
5 13522105 	
6	
	Tanda Tangan Asisten: 

- Pertemuan : gabisa lihat profil diri sendiri kalau diprint.

### Asistensi III

Tanggal : 23 November 2023	Catatan Asistensi:
Tempat : Lab Pemrograman	
Kehadiran Anggota Kelompok:	
No	
NIM	
Tanda tangan	
1	- Ubah kicauan golokch jingya cuma spani (samarin kaya buat kicau)
2	- Utas -1 segmentation fault.
3	- Utas, kicauan, badan cobain -1 semua.
4	- Collision hashtag → jlarin di laporan kalo ngs ditande.
5	- Draft reepint angka random.
6	- Draft kembali malah ketapus.
	- Draft terkait segfault.
	- feltyvi pertamanan reepint "manik"
	Tanda Tangan Asisten:





### 9.3 Log Activity Anggota Kelompok

Adril Putra Merin / 13522068

- 1 November 2023 : Menginisiasi struktur *file*
- 1 November 2023 : Memasukkan ADT Mesin Karakter dan Mesin Kata
- 2 November 2023 : Membuat Makefile
- 3 November 2023 : Memasukkan file user
- 3 November 2023 : Memperbarui README.md
- 4 November 2023 : Menambahkan pcolor
- 4 November 2023 : Membuat listuser
- 4 November 2023 : Memperbarui definisi readWord() pada ADT Mesin Kata
- 4 November 2023 : Membuat Login() pada user
- 4 November 2023 : Menambahkan app friend
- 4 November 2023 : Memperbarui driver untuk main dan wordmachine
- 15 November 2023 : Membuat ADT DSU
- 15 November 2023 : Menambahkan fungsi searchFriendGroup pada file user
- 17 November 2023 : Menambahkan fungsi untuk memeriksa word yang hanya mengandung spasi
- 17 November 2023 : Memperbaiki beberapa bug pada user
- 23 November 2023 : Menambahkan draft pada main
- 23 November 2023 : Memperbaiki bug pada friend, draft, dan tweet
- 24 November 2023 : Membuat ADT Matrix
- 24 November 2023 : Memperbaiki error pada listuser

Marvel Pangondian / 13522075

- 2 November 2023 : Menambah ADT Time, Datetime, dan Kicauan
- 2 November 2023 : Modifikasi Datetime
- 3 November 2023 : Menambah listlinear
- 3 November 2023 : Menambah tweet dan listkicauan
- 3 November 2023 : modifikasi prosedur bacaKicauan
- 5 November 2023 : Memperbarui tweet, listkicauan dan wordmachine

5 November 2023 : Menambahkan prosedur putusUtas dan insertUtas  
20 November 2023 : Debug dan memperbarui utas  
21 November 2023 : Menambah fungsi datetime\_to\_word dan string\_to\_datetime  
23 November 2023 : Menambahkan ADT List dengan Struktur Data Array Statik dan  
List dengan Struktur Data Array Dinamik  
24 November 2023 : Menulis laporan

Evelyn Yosiana / 13522083

10 November 2023 : Menambahkan reply, tree balasan, dan ADT Tree  
14 November 2023 : Memperbaiki binary tree dan menambahkan ADT List dengan  
Struktur Data Berkait  
14 November 2023 : Memperbaiki tree balasan  
16 November 2023 : Menambahkan nodeBalasan, linkedListBalasan, memperbaiki reply,  
dan memperbaiki treeBalasan  
21 November 2023 : Memperbaiki tree balasan  
21-24 November 2023 : Menulis laporan

Sa'ad Abdul Hakim / 13522092

24 November 2023 : Menambah prosedur save dan load  
24 November 2023 : Memperbaiki folder exist

Fabian Radenta Bangun / 13522105

22 November 2023 : Menambahkan ADT Stack dengan struktur data berkait  
22 November 2023 : Menambahkan draft  
24 November 2023 : Menambah ADT List dengan Struktur Data Array Statik dan Tree  
24 November 2023 : Menulis laporan