

LAPORAN TUGAS BESAR

IF2110 Algoritma dan Struktur Data

BurBir


Dipersiapkan oleh:

Kelompok H / K-03

Shabrina Maharani	13522134
Ikhwan Al Hakim	13522147
Muhammad Roihan	13522152
Rafif Ardhinto Ichwantoro	13522159
Mohammad Akmal Ramadan	13522161

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen	Halaman
		<i>IF2110-TB-H-K03</i>	<i><jml hlm></i>

		<i>Revisi</i>	<i><no revisi></i>	<i><Tgl release></i>
--	--	---------------	--------------------------	----------------------------

Daftar Isi

1	Ringkasan.....	5
2	Struktur Data (ADT).....	5
2.1	ADT Sederhana.....	5
2.2	ADT List dengan Struktur Data Array Statik.....	6
2.3	ADT Matriks.....	7
2.4	ADT List dengan Struktur Data Array Dinamis.....	8
2.5	Mesin Karakter dan Mesin Kata.....	9
2.6	ADT Priority Queue.....	10
2.7	ADT Stack.....	11
2.8	ADT List dengan Struktur Data Berkait.....	11
2.9	ADT Tree.....	12
2.10	ADT Graf dengan Representasi Adjacency Matrix.....	13
3	Program Utama.....	14
4	Data Test.....	17
4.1	Data Test 1.....	17
4.2	Data Test 2.....	17
4.3	Data Test 3.....	18
4.4	Data Test 4.....	19
4.5	Data Test 5.....	19
4.6	Data Test 6.....	20
4.7	Data Test 7.....	21
4.8	Data Test 8.....	21
4.9	Data Test 9.....	22
4.10	Data Test 10.....	22
4.11	Data Test 11.....	22
4.12	Data Test 12.....	23
4.13	Data Test 13.....	23
4.14	Data Test 14.....	23
4.15	Data Test 15.....	24
4.16	Data Test 16.....	24
4.17	Data Test 17.....	25
4.18	Data Test 18.....	26
4.19	Data Test 19.....	26
4.20	Data Test 20.....	26
4.21	Data Test 21.....	27
4.22	Data Test 22.....	28

4.23 Data Test 23.....	28
4.24 Data Test 24.....	29
4.25 Data Test 25.....	29
4.26 Data Test 26.....	30
4.27 Data Test 27.....	30
4.28 Data Test 28.....	31
4.29 Data Test 29.....	32
5 Test Script.....	32
6 Pembagian Kerja dalam Kelompok.....	39
7 Lampiran.....	40
7.1 Deskripsi Tugas Besar 1.....	40
7.2 Notulen Rapat.....	41
7.3 Log Activity Anggota Kelompok.....	42
7.4 Form Asistensi.....	43
7.5 Milestone.....	47

1 Ringkasan

BurBir (Burung Biru) merupakan program berbasis CLI yang mensimulasikan sebuah media sosial. Mulanya program ini terinisiasi untuk membantu Ande-Ande Lumut yang iba saat mendengar kisah Klenting Kuning yang sedih karena dia dirundung oleh ibu tirinya dan saudara tirinya di Kuning Space. BurBir dibuat semirip mungkin dengan sosial media yang marak pada saat itu.

Berdasarkan spesifikasi program, kami membuat program dengan 11 komponen diantaranya, yaitu inisiasi dari program BurBir, pembacaan perintah agar pengguna dapat berinteraksi dengan program CLI, pengguna dapat melakukan inisialisasi data pribadi pengguna, profil pengguna, melakukan pertemanan dengan pengguna lain, melakukan permintaan pertemanan, kicauan, membuat balasan, draf kicauan, membuat utas, dan melakukan simpan serta muat.

Laporan mencakup deskripsi umum dari program BurBir, penjelasan tambahan spesifikasi tugas, penjelasan struktur data yang digunakan, penjelasan program utama, algoritma-algoritma menarik yang ditemukan saat membuat program, data-data dan script yang digunakan untuk menguji keberjalanan program BurBir, pembagian tugas, dan lampiran lainnya yang berisi deskripsi tugas besar, notulensi rapat, *log activity* tiap anggota kelompok dan lampiran lainnya.

Program dibuat menggunakan bahasa C dengan memanfaatkan ADT yang telah dipelajari pada mata kuliah IF2110 Algoritma dan Struktur Data Semester 3 Tahun 2023/2024. Program ini dimulai dengan menampilkan inisiasi dengan pengguna memasukkan input berupa nama folder konfigurasi untuk dimuat. Setelah berhasil dimuat, pengguna akan diminta untuk melakukan daftar jika belum mempunyai akun dan masuk jika sudah. Pengguna juga dapat melakukan keluar untuk melakukan logout. Setelah masuk, pengguna dapat memberikan perintah seperti melengkapi profil, membuat kicauan dan hal lain yang berhubungan dengan kicauan, membuat utas, bersosialisasi dengan teman dengan balasan, dan membuat draf. Hasil kelompok kami, program sudah dapat berjalan sesuai dengan spesifikasi.

2 Struktur Data (ADT)

2.1 ADT Sederhana

- Sketsa struktur data datetime:

```
typedef struct
{
    int DD;    /* integer [1..31] */
    int MM;    /* integer [1..12] */
    int YYYY; /* integer [1900..2030] */
    TIME T;
} DATETIME;
```

DD : Tempat yang menyimpan informasi tanggal

MM : Tempat yang menyimpan informasi bulan

YYYY : Tempat yang menyimpan informasi tahun

T : Struktur data time

DATETIME merupakan sebuah *struct* yang berisi keempat elemen di atas

Sketsa struktur data time:

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

HH : Tempat yang menyimpan informasi jam

MM : Tempat yang menyimpan informasi menit

SS : Tempat yang menyimpan informasi detik

TIME merupakan sebuah *struct* yang berisi ketiga elemen di atas

- Persoalan yang diselesaikan:
ADT ini digunakan untuk menampilkan informasi tanggal pembuatan kicauan.
- Alasan pemilihan:
Dibanding ADT yang lain, ADT ini dapat merepresentasikan tanggal dalam bentuk yang sesuai format tanggal pada umumnya.
- Implementasi sebagai ADT (nama file)
Nama file: *datetime.h*, *datetime.c*

2.2 ADT List dengan Struktur Data Array Statik

- Sketsa struktur data:

```
typedef struct {
    Word nama;
    Word pass;
    Word bio;
    Word hp;
    Word weton;
    Word jakun;
    Matrix PP;
} Pengguna;

typedef struct {
    Pengguna db[20];
    int Neff;
} UserDB;
```

nama : Tempat yang menyimpan informasi nama pengguna

password : Tempat yang menyimpan informasi password dari suatu akun pengguna
 bio : Tempat yang menyimpan informasi bio dari suatu profil pengguna
 hp : Tempat yang menyimpan informasi dari nomor hp pengguna
 weton : Tempat yang menyimpan informasi weton pengguna
 jakun : Tempat yang menyimpan informasi status akun pengguna publik atau privat
 PP : Sebuah struktur matriks yang merepresentasikan profil pengguna
 Pengguna merupakan sebuah struct yang terdiri dari elemen di atas. Sedangkan db merupakan array yang berisi kumpulan pengguna dan mengimplementasikan ADT List dengan struktur data array statik.

- Persoalan yang diselesaikan :
 ADT ini membantu menyelesaikan semua fungsi yang berkaitan dengan pengguna seperti ganti profil, lihat profil, masuk, keluar ataupun hubungan antar sesama pengguna.
- Alasan pemilihan :
 Dikarenakan banyak sekali fungsi yang berkaitan dengan pengguna, jadi diperlukan sebuah ADT yang dapat menyimpan informasi - informasi yang berkaitan dengan pengguna.
- Implementasi sebagai ADT (nama file)
 Nama file: *pengguna.h*, *pengguna.c*

2.3 ADT Matriks

- Sketsa struktur data:

```
typedef int IdxType; /* Index baris/kolom */
typedef char ElType;
typedef struct
{
    ElType mem[ROW_CAP][COL_CAP];
    int rowEff; /* banyaknya/ukuran baris */
    int colEff; /* banyaknya/ukuran kolom */
} Matrix;
```

mem : Tempat penyimpanan elemen matrix
 rowEff : Banyaknya ukuran baris yang terdefinisi
 colEff : Banyaknya ukuran kolom yang terdefinisi
 Matrix merupakan sebuah struct yang berisi ketiga elemen di atas.

- Persoalan yang diselesaikan:
ADT ini digunakan untuk foto profil pengguna.
- Alasan pemilihan:
Dibanding ADT yang lain, ADT ini dapat merepresentasikan data dalam bentuk baris dan kolom yang mana ini sangat sesuai dengan bentuk foto profil.
- Implementasi sebagai ADT (nama file)
Nama file: *pengguna.c* (*bagian profil*)

2.4 ADT List dengan Struktur Data Array Dinamis

- Sketsa struktur data ListDinkicau:

```
typedef struct
{
    Kicau *buffer; /* memori tempat menyimpan elemen
    int nEff;      /* >=0, banyaknya elemen efektif
    int capacity;  /* ukuran elemen */
} ListDinkicau;
```

*buffer : memori tempat menyimpan elemen yang bertipe Kicau.
neff : tipe data integer yang merupakan banyaknya elemen efektif.
capacity : tipe data integer yang menyatakan ukuran array dinamis.
ListDinkicau : merupakan list dengan struktur data array dinamis yang memiliki berisi ketiga elemen di atas.

Sketsa struktur Kicau:

```
// Definisi tipe data kicaun
typedef struct {
    int id;
    Word text;
    int like;
    Word author;
    Word date;
    Word jakunkicau;
} Kicau;
```

id : Tempat yang menyimpan informasi id kicaun
text : Tempat yang menyimpan informasi isi kicaun
like : Tempat yang menyimpan informasi jumlah like dari suatu kicaun
author : Tempat yang menyimpan informasi nama pembuat kicaun
date : Tempat yang menyimpan informasi tanggal kicaun diterbitkan
jakunkicau : Tempat yang menyimpan informasi status akun pembuat kicaun
Kicau merupakan sebuah *struct* yang berisi keenam elemen di atas

- Persoalan yang diselesaikan :
ADT ini digunakan dalam fungsi kicau (kicau.h dan kicau.c) untuk menyimpan kicauan user yang jumlahnya dapat terus bertambah tanpa maksimal.
- Alasan pemilihan :
Untuk menyimpan kicauan - kicauan dari user agar data mudah dimanipulasi, ADT ini sangat cocok dibandingkan dengan ADT lainnya. contoh jika menggunakan struktur data array statik akan mengalami kepenruhan data disebabkan tidak ada batas ketentuan user melakukan kicau sehingga digunakan ADT ini agar memori tidak terbatas.
- Implementasi sebagai ADT (nama file)
Nama file: *kicauan.c* *kicauan.h*

2.5 Mesin Karakter dan Mesin Kata

- Sketsa struktur data:

```
char currentChar;
boolean EOP;

static FILE *pita;
static int retval;
```

Mesin karakter akan membaca stream dalam bentuk pita. Karakter yang saat ini dibaca disimpan dalam currentChar.

```
typedef struct
{
    char TabWord[NMax];
    int Length;
} Word;
```

Dengan memanfaatkan mesin karakter sebagai input, setiap karakter yang telah dibaca pada mesin karakter akan disimpan kedalam TabWord. Length merupakan jumlah karakter yang disimpan pada TabWord.

- Persoalan yang diselesaikan:
ADT ini digunakan untuk semua fungsi yang memerlukan input dari user. Selain itu ADT ini juga berguna untuk membuat nama, password, bio, no hp, weton dan jakun pada profil.
- Alasan pemilihan:
ADT ini digunakan semua fungsi yang memerlukan input dari user karena spesifikasi tubes kali ini mewajibkan segala masukan menggunakan mesin kata.

Dikarenakan nama, password, bio, no hp, weton dan jakun memerlukan input dari user dan tidak perlu direpresentasikan dalam bentuk matrix seperti foto profil, maka ADT ini sangat cocok untuk membuat hal hal tersebut.

- Implementasi sebagai ADT (nama file)
Nama file: *wordmachine.h*, *wordmachine.c*

2.6 ADT Priority Queue

- Sketsa struktur data:

```
typedef struct
{
    /* data */
    int IDpengirim;
    int IDpenerima;
    int Jumlahteman;
}teman;

typedef int addressPrioqueue; /*indeks tabel*/

typedef struct {
    teman *T;
    addressPrioqueue HEADQ;
    addressPrioqueue TAILQ;
    int MaxElQ;
}prioqueuefren;
/* Definisi PrioQueueFren kosong: HEAD=Nil; TAIL=Nil. */
```

addressPrioqueue: Alias untuk tipe data integer yang digunakan sebagai indeks untuk tabel dalam priority queue.

HEADQ dan TAILQ bertindak sebagai penunjuk ke elemen pertama dan terakhir dalam queue.

MaxElQ adalah variabel yang menyimpan kapasitas maksimum dari priority queue.

*T adalah pointer yang menunjuk ke array dari struktur teman yang akan menyimpan elemen-elemen priority queue.

- Persoalan yang diselesaikan:
ADT ini digunakan dalam fungsi pertemanan (*pertemanan.c* dan *pertemanan.h*) untuk mengurutkan elemen queue berdasarkan popularitas.
- Alasan pemilihan:
Dibandingkan dengan ADT lainnya, ADT prioqueue memiliki keunggulan dalam mengurutkan elemen di dalamnya berdasarkan suatu kriteria, yang mana ini sangat berguna untuk fungsi pertemanan yang membutuhkan suatu ADT untuk mengurutkan elemen-elemen queue berdasarkan kepopularitasan.
- Implementasi sebagai ADT (nama file)
Nama file: *pertemanan.c*, *pertemanan.h*

2.7 ADT Stack

- Sketsa struktur data:

```
typedef struct {
    Kicau T[MaxEl]; /* tabel penyimpan elemen */
    address TOP; /* alamat TOP: elemen puncak */
    Word author;
    int Nstack; /* jumlah stak */
} Stack;

typedef struct
{
    Stack *buffer; /* memori tempat penyimpan elemen (container)
    int nEff; /* >=0, banyaknya elemen efektif */
    int capacity; /* ukuran elemen */
} ListStack;
```

ListStack : listdin yang berelemen stack.

tipe data Stack berisikan:

TOP : merupakan pointer ke elemen puncak.

author : merupakan word yang mendefinisikan pemilik dari stack.

Nstack : tipe data integer yang menunjukkan jumlah tumpukan.

Kicau: tipe data kicau sebagai penyimpanan draft.

- Persoalan yang diselesaikan:
ADT ini digunakan dalam fungsi draftkicauan (draftkicauan.h dan draftkicauan.c) untuk menyimpan draft dalam bentuk stack.
- Alasan pemilihan:
ADT stack dipilih karena sesuai spesifikasi algoritma penyimpanan draft yang menggunakan sistem LIFO(*last in first out*). contohnya pada fungsi LIHAT_DRAFT ketika dijalankan akan menampilkan top dari stack.
- Implementasi sebagai ADT (nama file)
Nama file: *draftkicauan.c draftkicauan.h*

2.8 ADT List dengan Struktur Data Berkait

- Sketsa Struktur data :

```

typedef struct utas* Address;
✓ typedef struct utas {
    Word isi;
    Word date;
    Address next;
} Utas;

✓ typedef struct {
    Kicau k;
    Address u;
} KicauUtas;

```

isi : Tempat yang menyimpan informasi isi dari sebuah kicau yang menjadi elemen utas

date : Tempat yang menyimpan informasi tanggal dibentuknya suatu kicau yang merupakan elemen utas

Utas sendiri merupakan sebuah struktur yang berisi kedua elemen di atas. Sedangkan KicauUtas merupakan sebuah struktur yang mengimplementasikan ADT list dengan struktur data statik yang berisi kicau kicau yang menyambung membentuk utas dan Address yang menunjuk kicau yang menjadi elemen dari suatu utas.

- Persoalan yang diselesaikan:
ADT ini digunakan dalam fungsi utas (*utas.c* dan *utas.h*) untuk menghubungkan kicauan sambungan dengan kicauan utama dalam sebuah utas.
- Alasan pemilihan:
Dibandingkan dengan ADT lainnya, ADT list dengan struktur data berkait memiliki keunggulan dalam menghubungkan elemen yang satu dengan elemen lainnya sehingga menjadi satu kesatuan, yang mana ini sangat berguna untuk fungsi utas yang membutuhkan suatu ADT untuk menghubungkan elemen-elemen menjadi satu kesatuan utas.
- Implementasi sebagai ADT (nama file)
Nama file: *utas.c* *utas.h*

2.9 ADT Tree

- Sketsa struktur data:

```

typedef Balasan InfotypeBalasan;
typedef struct TNodeBalasan *AddressBalasan;
typedef struct TNodeBalasan
{
    InfotypeBalasan T;
    AddressBalasan *SubTree; // Tree* Subtree (array
    int Count;
    int Capacity;
    int IDParent;
} NodeBalasan;

typedef struct {
    AddressBalasan *isi;
    int neff;
} ListTreeBalasan;
// Tree == Address == [0..n] of pointers to TNode

```

Count : Tempat yang menyimpan informasi jumlah balasan
 Capacity : Tempat yang menyimpan informasi kapasitas maksimum balasan
 IDParent : Tempat yang menyimpan informasi IDParent untuk balasan
 Node Balasan merupakan sebuah struktur yang berisi ketiga elemen di atas
 isi : Tempat yang menyimpan informasi isi balasan
 neff : Tempat yang menyimpan informasi jumlah memori efektif
 ListTreeBalasan merupakan sebuah struktur yang berisi kumpulan seluruh balasan

- Persoalan yang diselesaikan :
 ADT ini digunakan dalam penyelesaian fungsi balasan, ADT terdapat pada file balasan.h, digunakan dalam balasan.c .
- Alasan pemilihan :
 Spesifikasi fungsi balasan dengan penyimpanan memori dan cara kerja. setiap balasan bisa dibalas dan lebih dari satu balasan , algoritma ini cocok dengan ADT yang kita pilih.
- Implementasi sebagai ADT (nama file)
 Nama file: *balasan.c* *balasan.h*

2.10 ADT Graf dengan Representasi Adjacency Matrix

- Sketsa struktur data:

```

typedef struct
{
    ElGraf mem[ROW_GRAF][COL_GRAF];
} Graf;

typedef struct {
    EllistGraf content[CAPACITYGRAF]; /* memori tempat penyimpan elemen (container) */
} ListGraf;

```

Graf adalah sebuah struktur yang memiliki sebuah matriks dua dimensi dengan tipe ElGraf.

mem merupakan matriks dengan ROW_GRAF baris dan COL_GRAF kolom yang menyimpan elemen-elemen bertipe ElGraf.

ListGraf adalah sebuah struktur yang memiliki array content dengan tipe EllistGraf.

content adalah array dengan kapasitas maksimum CAPACITYGRAF yang digunakan sebagai kontainer untuk menyimpan elemen-elemen EllistGraf.

- Persoalan yang diselesaikan:
ADT Graf dengan representasi adjacency matrix digunakan dalam file *pertemanan.h*. Dengan graf, setiap pengguna digambarkan dengan simpul dan hubungan pertemanan dengan sebuah sisi.
- Alasan pemilihan:
Dibandingkan dengan ADT lainnya, ADT graf ini merupakan ADT yang paling cocok karena menggambarkan representasi yang lebih jelas dari list.
- Implementasi sebagai ADT (nama file)
Nama file: *graph.h graph.c*

3 Program Utama

Pada program utama, program akan memunculkan sebuah inisialisasi, lalu program akan meminta user untuk menginput folder konfigurasi yang akan dimuat. Lalu program akan melakukan muat untuk membaca file konfigurasi. Jika file konfigurasi berhasil dimuat, pengguna dapat memberikan perintah untuk melakukan administrasi data seperti daftar jika belum memiliki akun dan masuk jika sudah memiliki akun. Pengguna juga dapat melakukan tutup program dan keluar dengan kondisi pengguna sudah masuk.

Kemudian, saat program sudah dalam kondisi masuk, program dapat menerima beberapa command yang valid seperti, 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, UBAH_KICAUAN, BALAS, BALASAN, HAPUS_BALASAN, BUAT_DRAF, LIHAT_DRAF, UTAS, SAMBUNG_UTAS, HAPUS_UTAS, CETAK_UTAS, SIMPAN, dan MUAT.

Masukan yang tidak valid akan membuat program memberikan pesan yang menyatakan kesalahan dan meminta input kembali sampai input dinyatakan sesuai. Berikut penjelasan setiap perintah yang dapat dijalankan:

- DAFTAR

Fungsi program yang dapat membuat pengguna dapat mendaftar dan terdaftar dengan nama unik beserta password antar pengguna.

- MASUK

Fungsi program yang dapat membuat pengguna berhasil masuk dan menjalankan perintah lain dengan masukan.

- **KELUAR**
Fungsi program yang dapat membuat pengguna dapat keluar dari akun jika sebelumnya pernah login.
- **TUTUP_PROGRAM**
Fungsi program yang dapat membuat pengguna dapat keluar dari program BurBir.
- **GANTI_PROFIL**
Fungsi program yang dapat membuat pengguna dapat mengganti profil.
- **LIHAT_PROFIL**
Fungsi program yang dapat membuat pengguna dapat melihat profil orang lain.
- **ATUR_JENIS_AKUN**
Fungsi program yang dapat membuat pengguna dapat mengganti akunnya.
- **UBAH_FOTO_PROFIL**
Fungsi program yang dapat membuat pengguna dapat mengubah foto profil .
- **DAFTAR_TEMAN**
Fungsi program yang dapat menampilkan daftar teman yang dimiliki oleh pengguna.
- **HAPUS_TEMAN**
Fungsi program yang dapat membuat pengguna bisa menghapus pertemanan yang dimiliki pengguna dengan seorang teman.
- **TAMBAH_TEMAN**
Fungsi program yang dapat membuat pengguna dapat menambahkan teman.
- **DAFTAR_PERMINTAAN_PERTEMANAN**
Fungsi program yang dapat membuat output dan menampilkan daftar permintaan pertemanan yang dimiliki pengguna.
- **SETUJUI_PERTEMANAN**
Fungsi program yang dapat membuat pengguna bisa menyetujui pertemanan jika pengguna lain melakukan permintaan pertemanan.
- **KICAU**
Fungsi program yang dapat membuat pengguna bisa membuat sebuah kicau dan ditampilkan di layar.
- **KICAUAN**
Fungsi program yang dapat menampilkan kicau dari pengguna itu sendiri dan kicau yang dibuat oleh teman pengguna.
- **SUKA_KICAUAN**
Fungsi program yang dapat membuat pengguna bisa melakukan suka pada suatu kicau dengan id tertentu.
- **UBAH_KICAUAN**

Fungsi program yang dapat membuat pengguna bisa melakukan suatu perubahan isi kicau pada suatu kicau dengan id tertentu.

- **BALAS**

Fungsi program yang dapat membuat pengguna bisa membalas kicau yang dibuat oleh pengguna (*author*) lain dengan id kicau tertentu.

- **BALASAN**

Fungsi program yang dapat membuat pengguna bisa melihat daftar balasan yang dibalas oleh pengguna lain pada kicau yang dibuat oleh pengguna tersebut.

- **HAPUS_BALASAN**

Fungsi program yang dapat membuat pengguna bisa menghapus balasan dengan IDBalasan tertentu dan balasan yang dapat dihapus hanya balasan yang dibuat oleh pengguna itu sendiri.

- **BUAT_DRAF**

Fungsi program yang dapat membuat pengguna bisa membuat draf dan draf tersebut bisa disimpan dengan perintah SIMPAN, dihapus dengan perintah HAPUS, dan diterbitkan menjadi kicau dengan perintah TERBIT.

- **LIHAT_DRAF.**

Fungsi program yang dapat membuat pengguna bisa melihat draf terakhir yang dibuat dan akan melakukan validasi apakah draf ada atau tidak. LIHAT_DRAF juga bisa menjalankan HAPUS,SIMPAN,dan TERBIT seperti BUAT_DRAF.

- **UTAS**

Fungsi program yang dapat membuat pengguna dapat membuat utas dengan kicau yang dirangkai dengan adanya kicauan utama dan kicauan sambungan. Jika utas sudah dibuat maka utas akan memiliki IDUtas.

- **SAMBUNG_UTAS**

Fungsi program yang dapat membuat pengguna bisa menambah sebuah kicauan baru dalam sebuah utas dengan id=IDUtas pada suatu posisi indeks tertentu.

- **HAPUS_UTAS**

Fungsi program yang dapat membuat pengguna bisa menghapus kicauan dalam sebuah utas yang mempunyai IDUtas tertentu dengan memasukkan IDKicau yang ingin dihapus.

- **CETAK_UTAS**

Fungsi program yang dapat membuat output dan menampilkan ke layar keseluruhan kicauan dalam sebuah utas dengan IDUtas tertentu.

- **SIMPAN**

Fungsi yang dapat membuat pengguna bisa menyimpan kondisi dari program BurBir dengan keadaan saat ini ke dalam suatu folder berkas-berkas konfigurasi.

- **MUAT**

Fungsi yang dapat membuat pengguna bisa memuat kondisi dari folder berisi berkas-berkas konfigurasi ke dalam aplikasi BurBir dan hanya dapat dilakukan sebelum pengguna menginput perintah masuk.

4 Data Test

Untuk memvalidasi program berjalan dengan tepat dari program, pengujian diperlukan untuk setiap fungsi program. Berikut adalah semua fungsi utama yang diuji, bersama dengan rincian, hasil keluaran yang diharapkan, dan data pengujian yang digunakan untuk masing-masing fungsi.

4.1 Data Test 1

Memeriksa apakah program dapat dimulai dan dapat membaca file config. Setelah memasukkan nama file diharapkan config program berhasil berjalan.



```

  BURBIR

Selamat datang di BurBir!
Silahkan masukkan nama folder konfigurasi: config-1;

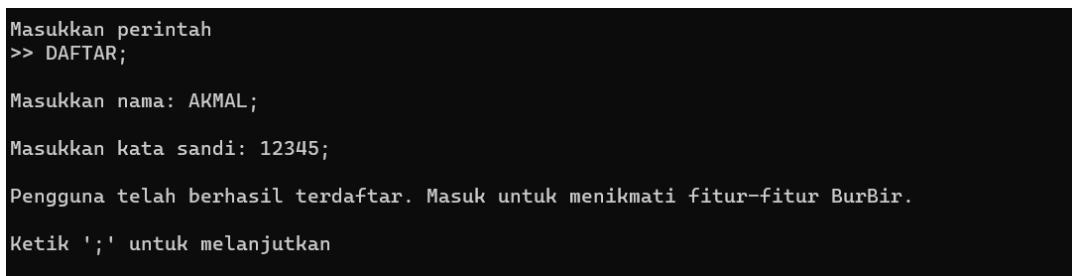
File konfigurasi berhasil dimuat! Selamat menggunakan aplikasi BurBir!
Ketik ';' untuk melanjutkan

```

Gambar 4.1.1 Output ketika inisialisasi

4.2 Data Test 2

Memeriksa apakah pengguna mendaftar dan terdaftar dengan nama unik beserta password antar pengguna. Setelah memasukkan nama dan password, diharapkan pengguna berhasil terdaftar.



```

Masukkan perintah
>> DAFTAR;

Masukkan nama: AKMAL;

Masukkan kata sandi: 12345;

Pengguna telah berhasil terdaftar. Masuk untuk menikmati fitur-fitur BurBir.

Ketik ';' untuk melanjutkan

```

Gambar 4.2.1 Output berhasil daftar

```

Masukkan perintah
>> DAFTAR;

Masukkan nama: AKMAL;

Wah, sayang sekali nama tersebut telah diambil.

Masukkan nama: |

```

Gambar 4.2.2 Kondisi ketika nama telah diambil

4.3 Data Test 3

Memeriksa apakah pengguna berhasil masuk dan menjalankan perintah lain dengan masukan nama dan password yang sesuai. Setelah memasukkan nama dan password, diharapkan pengguna berhasil masuk apabila memasukkan nama dan password dengan benar dan pengguna belum login.

```

Masukkan perintah
>> MASUK;

Masukkan nama: AKMAL;

Masukkan kata sandi: 12345;

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

Ketik ';' untuk melanjutkan
|

```

Gambar 4.3.1 Kondisi ketika berhasil masuk

```

Masukkan perintah
>> MASUK;

Masukkan nama: AKMAL;

Masukkan kata sandi: 1234;

Wah, kata sandi yang Anda masukkan belum tepat. Periksa kembali kata sandi Anda!

Masukkan kata sandi: |

```

Gambar 4.3.2 Kondisi ketika password salah

```

Masukkan perintah
>> MASUK;

Masukkan nama: AKMA;

Wah, nama yang Anda cari tidak ada. Masukkan nama lain!

Masukkan nama: |

```

Gambar 4.3.3 Kondisi ketika nama salah

4.4 Data Test 4

Memeriksa apakah pengguna berhasil keluar jika sebelumnya pernah login. Setelah mengetik *command* KELUAR diharapkan pengguna berhasil keluar apabila pengguna sudah login.

```

Masukkan perintah
>> KELUAR;

Anda berhasil logout. Sampai jumpa di pertemuan berikutnya!

Ketik ';' untuk melanjutkan

|

```

Gambar 4.4.1 Kondisi ketika berhasil keluar

```

Masukkan perintah
>> KELUAR;

Anda belum login! Masuk terlebih dahulu untuk menikmati layanan BurBir.

Ketik ';' untuk melanjutkan

|

```

Gambar 4.4.2 Kondisi ketika gagal keluar karena belum login

4.5 Data Test 5

Memeriksa apakah pengguna dapat keluar dari program BurBir. Setelah megetik *command* TUTUP_PROGRAM, diharapkan pengguna berhasil keluar dari program BurBir.

```
Masukkan perintah
>> TUTUP_PROGRAM;

Anda telah keluar dari program BurBir. Sampai jumpa di penjelajahan berikutnya.
```

Gambar 4.5.1 Kondisi ketika berhasil menutup program

4.6 Data Test 6

Memeriksa apakah pengguna berhasil mengganti profil. Setelah memasukkan profil yang baru, diharapkan profil pengguna akan berganti dengan profil yang baru dimasukkan.

```
Masukkan perintah
>> GANTI_PROFIL;

| Nama: AKMAL
| Bio Akun: BUBIR
| No HP: 08229912344
| Weton: Pahing

Masukkan Bio Akun: BUBITY;

Masukkan No HP: ASD;
Format no Hp salah!

Masukkan No HP: 0822117544;

Masukkan Weton: PAHIG;
Masukkan weton salah

Masukkan Weton: Pahing;

Profil Anda sudah berhasil diperbarui!

Ketik ';' untuk melanjutkan
```

Gambar 4.6.1 Kondisi ketika berhasil ganti profil

[illegible]

Gambar 4.6.2 Kondisi ketika bio akun melebihi 135 karakter

4.7 Data Test 7

Memeriksa apakah pengguna dapat melihat profil orang lain. Setelah memasukkan nama pengguna yang ingin dilihat, diharapkan output akan menampilkan profil pengguna tersebut apabila berjenis akun publik atau telah berteman dengan pengguna.

```
Masukkan perintah
>> LIHAT_PROFIL AKMAL;

Wah, akun AKMAL diprivat nih. Ikuti dulu yuk untuk bisa melihat profil AKMAL!

Ketik ';' untuk melanjutkan
```

Gambar 4.7.1 Kondisi ketika gagal melihat profil

```
Masukkan perintah
>> LIHAT_PROFIL Tuan Hak;
| Nama: Tuan Hak
| Bio Akun: Ini bio lho
| No HP: 08
| Weton: Pahing

Foto profil:
*****
*Q*Q*
*Q*Q*
*Q*Q*
*Q*Q*
*****

Ketik ';' untuk melanjutkan
```

Gambar 4.7.2 Kondisi ketika berhasil melihat profil

4.8 Data Test 8

Memeriksa apakah pengguna dapat mengganti akunnya. Setelah mengetik YA, diharapkan jenis akun pengguna berganti, namun jika mengetik TIDAK jenis akun pengguna tidak berganti.

```
Masukkan perintah
>> ATUR_JENIS_AKUN;

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

Akun anda sudah diubah menjadi akun Publik.

Ketik ';' untuk melanjutkan
```

Gambar 4.8.1 Kondisi ketika berhasil mengubah jenis akun

4.9 Data Test 9

Memeriksa apakah pengguna dapat mengubah foto profil. Setelah memasukkan foto profil baru, diharapkan foto profil pengguna akan berganti sesuai dengan yang baru dimasukkan.

```
>> UBAH_FOTO_PROFIL;  
Foto profil anda saat ini:  
*****  
*****  
*****  
*****  
*****  
  
Masukkan Foto Profil Baru  
  
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 *;  
Foto profil anda sudah berhasil diganti!  
  
Ketik ';' untuk melanjutkan
```

Gambar 4.9.1 Kondisi ketika berhasil mengubah foto profil

4.10 Data Test 10

Memeriksa output apakah dapat menampilkan daftar teman yang dimiliki pengguna. Setelah mengetik *command* DAFTAR_TEMAN, diharapkan program akan menampilkan daftar teman yang dimiliki pengguna.

```
Masukkan perintah  
>> DAFTAR_TEMAN;  
Tuan Hak memiliki 2 teman.  
Daftar teman Tuan Hak:  
| Tuan Bri  
| Tuan Bus  
  
Ketik ';' untuk melanjutkan
```

Gambar 4.10.1 Kondisi ketika berhasil melihat daftar teman

4.11 Data Test 11

Memeriksa apakah dapat menghapus teman yang dimiliki pengguna. Setelah memasukkan nama teman dan pengguna mengetik YA, diharapkan program akan menghapus teman apabila sebelumnya pengguna saling berteman, Namun jika pengguna mengetik TIDAK program akan batal menghapus teman.

```

Masukkan perintah
>> HAPUS_TEMAN;
Masukkan nama pengguna:
Tuan Bri;
Apakah anda yakin ingin menghapus Tuan Bri dari daftar teman Anda? (YA/TIDAK)YA;
Tuan Bri berhasil dihapus dari daftar teman Anda.

Ketik ';' untuk melanjutkan

```

Gambar 4.11.1 Kondisi ketika berhasil hapus teman

4.12 Data Test 12

Memeriksa apakah pengguna dapat menambahkan teman. Setelah memasukkan nama pengguna, diharapkan program akan mengirimkan permintaan pertemanan kepada pengguna tersebut apabila pengguna ditemukan.

```

>> TAMBAH_TEMAN;
Terdapat permintaan pertemanan yang belum Anda setujui. Silakan kosongkan daftar permintaan pertemanan untuk Anda terlebih dahulu.

```

Gambar 4.12.1 Kondisi gagal tambah teman karena masih ada permintaan pertemanan yang belum disetujui

4.13 Data Test 13

Memeriksa apakah output dapat menampilkan permintaan teman yang dimiliki pengguna. Setelah mengetik *command* DAFTAR_PERMINTAAN_PERTEMANAN, diharapkan program akan menampilkan daftar permintaan teman yang dimiliki pengguna berdasarkan kepopuleran

```

Masukkan perintah
>> DAFTAR_PERMINTAAN_PERTEMANAN;
Terdapat 2 permintaan pertemanan untuk Anda.
| Tuan Vin

| Jumlah Teman: 0

| Tuan Bus

| Jumlah Teman: 1

Ketik ';' untuk melanjutkan

```

Gambar 4.13.1 Kondisi ketika berhasil melihat daftar permintaan teman

4.14 Data Test 14

Memeriksa apakah pengguna dapat menyetujui permintaan pertemanan. Setelah mengetik *command* SETUJUI_PERTEMANAN dan pengguna mengetik YA maka pengguna berhasil menjalin pertemanan sebaliknya jika pengguna mengetik TIDAK.

```

Masukkan perintah
>> SETUJUI_PERTEMANAN;
Permintaan pertemanan teratas dari Tuan Vin

| Tuan Vin

| Jumlah teman: 0
Apakah Anda ingin menyetujui permintaan pertemanan ini? (YA/TIDAK) YA;
Permintaan pertemanan dari Tuan Vin telah disetujui. Selamat! Anda telah berteman dengan Tuan Vin.

Ketik ';' untuk melanjutkan

```

Gambar 4.14.1 Kondisi ketika berhasil menyetujui pertemanan

4.15 Data Test 15

Memeriksa apakah pengguna dapat melakukan kicau. Setelah memasukkan kicauan, program akan mengirimkan detail kicauan

```

Masukkan perintah
>> KICAU;
Ketik kicauan Anda: TES KICAU;
Kicauan Anda berhasil ditambahkan!
| ID = 4
| Tuan Hak
| 24/11/2023 17:45:4
| TES KICAU
| Disukai: 0

Ketik ';' untuk melanjutkan

```

Gambar 4.15.1 Kondisi ketika berhasil kicau

4.16 Data Test 16

Memeriksa apakah output dapat menampilkan kicauan dari pengguna ataupun dari teman pengguna. Setelah mengetik *command* KICAUAN program akan menampilkan detail kicauan dari pengguna ataupun dari teman pengguna.


```

Masukkan perintah
>> KICAUAN;
| ID = 1
| Tuan Bus
| 14/10/2023 11:09:18
| Halooo
| Disukai: 12

| ID = 3
| Tuan Hak
| 14/10/2023 11:10:1
| anjay mabar
| Disukai: 18

| ID = 4
| Tuan Hak
| 24/11/2023 17:45:4
| TES KICAU
| Disukai: 0

```

Gambar 4.16.1 Kondisi ketika berhasil melihat kicauan

4.17 Data Test 17

Memeriksa apakah pengguna dapat menyukai kicauan berdasarkan ID kicau. Setelah memasukkan ID kicau, pengguna berhasil menyukai kicauan apabila ID kicau ditemukan.

```

Masukkan perintah
>> SUKA_KICAUAN 3;
Selamat! kicauan telah disukai!
Detil kicauan:
| ID = 3
| Tuan Hak
| 14/10/2023 11:10:1
| anjay mabar
| Disukai: 19

Ketik ';' untuk melanjutkan

```

Gambar 4.17.1 Kondisi ketika berhasil suka kicauan

```

>> SUKA_KICAUAN 4;
Tidak ada kicauan dengan id tersebut.

```

Gambar 4.17.2 Kondisi ketika tidak ada kicauan dengan id tersebut

4.18 Data Test 18

Memeriksa apakah pengguna dapat mengubah kicauan berdasarkan ID kicau. Setelah memasukkan kicauan baru, kicauan berhasil diganti dengan kicauan yang baru dimasukkan apabila ID kicau ditemukan.

```
Masukkan perintah  
>> UBAH_KICAUAN 2;|
```

Gambar 18.1

4.19 Data Test 19

Memeriksa apakah dapat Menambah balasan pada balasan dengan IDBalasan pada kicauan dengan IDKicau. Setelah input id kicau dan id balasan, jika id kicau dan id balasan valid maka balas akan ditambahkan ke id yang dituju.

```
Masukkan perintah  
>> BALAS 3 -1;  
Masukkan balasan:  
HALO PAK;  
Selamat! Balasan telah diterbitkan  
Detik balasan:  
| ID = 2  
| Tuan Hak  
| 24/11/2023 18:26:3  
| HALO PAK  
  
Ketik ';' untuk melanjutkan
```

Gambar 4.19.1 Kondisi berhasil membuat balasan

```
>> BALAS 3 10;  
Kicauan tersebut tidak memiliki balasan dengan id 10.  
  
Ketik ';' untuk melanjutkan
```

Gambar 4.19.2 Kondisi ketika id balasan tidak ditemukan

4.20 Data Test 20

Memeriksa apakah output dapat menampilkan daftar balasan pada kicau. Setelah memasukan id kicau, jika id valid akan ditampilkan seluruh balasan dalam sesuai kicauan tersebut menampilkan dengan sistem print DFS

```
>> BALASAN 1;
Kicauan dengan id tersebut tidak memiliki balasan.
```

Gambar 4.20.1 Kondisi ketika id tidak memiliki balasan

```
>> BALASAN 3;
| ID = 3
| Tuan Hak
| 14/10/2023 11:10:1
| anjay mabar
| Disukai: 18
|   | ID = 1
|   | Tuan Bri
|   | 14/10/2023 11:09:18
|   | Ini Balasan dari Node Utama, yaitu Kicauan ke-2
|   |   | ID = 2
|   |   | Tuan Man
|   |   | 14/10/2023 11:09:12
|   |   | Ini Balasan dari Balasan ID ke-1
|   |   |   | ID = 4
|   |   |   | Tuan Man
|   |   |   | 14/10/2023 11:09:1
|   |   |   | Ini Balasan dari Balasan ID ke-2
|   |   | ID = 3
|   |   | Tuan Man
|   |   | 14/10/2023 11:09:40
|   |   | Ini Balasan dari Balasan ID ke-1 tapi balasan yg kedua anjay
```

Gambar 4.20.2 Kondisi ketika berhasil menampilkan balasan

4.21 Data Test 21

Memeriksa apakah pengguna dapat menghapus balasan. Setelah memasukan id kicau dan id balasan, jika valid maka seluruh balasan dengan id yang dipilih akan terhapus.

```
Masukkan perintah
>> HAPUS_BALASAN 3 2;
Balasan berhasil dihapus.
```

Gambar 4.21.1 kondisi ketika balasan berhasil dihapus

```

| ID = 3
| Tuan Hak
| 14/10/2023 11:10:1
| anjay mabar
| Disukai: 18
|   | ID = 1
|   | Tuan Bri
|   | 14/10/2023 11:09:18
|   | Ini Balasan dari Node Utama, yaitu Kicauan ke-2
|   |   | ID = 3
|   |   | Tuan Man
|   |   | 14/10/2023 11:09:40
|   |   | Ini Balasan dari Balasan ID ke-1 tapi balasan yg kedua anjay

```

Gambar 4.21.2 balasan ketika berhasil dihapus

4.22 Data Test 22

Memeriksa apakah pengguna dapat membuat draf. user memasukan isi draf, lalu memilih perintah, jika memilih simpan draf akan disimpan dalam stack, jika memilih terbit maka draf akan langsung masuk ke kicau, jika memilih hapus tidak terjadi apa dan draf tidak tersimpan.

```

>> BUAT_DRAFT;
Masukan draf: tuan hak suka tidur;
Apakah anda ingin menghapus, menyimpan, atau menerbitkan draf ini?
SIMPAN;
masuk kesini
Draf telah berhasil disimpan!

Ketik ';' untuk melanjutkan

```

Gambar 4.22.1 Kondisi ketika berhasil membuat dan menyimpan draf

```

Masukkan perintah
>> LIHAT_DRAFT;
Ini draf terakhir anda:
indexnya 2
| 24/11/2023 18:11:1
| tuan hak suka tidur
Apakah anda ingin mengubah, menghapus, atau menerbitkan draf ini? (KEMBALI jika ingin kembali

```

Gambar 4.22.2 Bukti draf berhasil dibuat

4.23 Data Test 23

Memeriksa apakah output dapat menampilkan draf yang dimiliki pengguna. saat mulai fungsi, jika user memiliki draf maka top dari stack akan ditampilkan, draf dapat diubah, simpan dan dihapus.

```

Masukkan perintah
>> LIHAT_DRAFT;
Ini draf terakhir anda:
indexnya 2
| 14/10/2023 11:09:18
| Hehe 3
Apakah anda ingin mengubah, menghapus, atau menerbitkan draf ini? (KEMBALI jika ingin kembali)
UBAH;
Masukan draft yang baru:
Saya sedang senang;
Apakah anda ingin menghapus, menyimpan, atau menerbitkan draf ini?
TERBIT;
Kicauan Anda berhasil ditambahkan!
| ID = 4
| Tuan Hak
| 24/11/2023 18:09:1
| Saya sedang senang
| Disukai: 0

Ketik ';' untuk melanjutkan

```

Gambar 4.23.1 Kondisi ketika berhasil melihat, mengubah, dan menerbitkan draft.

```

>> LIHAT_DRAFT;
/ah, anda belum memiliki draf apapun! Buat dulu ya :D

```

Gambar 4.23.2 Kondisi ketika user belum, memiliki draf.

4.24 Data Test 24

Memeriksa apakah pengguna dapat membuat UTAS. Setelah memasukkan input yang diperlukan, utas baru akan berhasil dibuat dengan kicauan pertamanya adalah kicauan dengan ID yang dimasukkan apabila ID kicau ditemukan.

```

Masukkan perintah
>> UTAS 3;
Utas berhasil dibuat!
Masukkan kicauan:
Pagi yang cerah
;
Apakah anda ingin melanjutkan utas ini? (Y/N)
Y;
Masukkan kicauan:
saya ingin kopi;
Apakah anda ingin melanjutkan utas ini? (Y/N)
N
;
Utas selesai!

Ketik ':' untuk melanjutkan

```

Gambar 4.24.1 Kondisi ketika berhasil membuat utas

```

>> UTAS 4;
Tidak ada kicauan yang memiliki id berikut.

```

Gambar 4.24.2 Kondisi ketika utas tidak ditemukan

4.25 Data Test 25

Memeriksa apakah pengguna dapat menambahkan kicauan baru dalam sebuah utas dengan ID Utas dan pada posisi index. Setelah memasukkan ID utas dan index, sebuah kicauan baru berhasil ditambahkan dalam sebuah utas apabila ID utas ditemukan dan index valid.

```
Masukkan perintah  
>> SAMBUNG_UTAS 3 2;  
Masukkan kicauan:  
INI SAMBUGNA;  
Kicauan berhasil disambung!  
  
Ketik ';' untuk melanjutkan
```

Gambar 4.25.1 Kondisi ketika berhasil menyambung utas

```
>> SAMBUNG_UTAS 4 1;  
Utas tidak ditemukan!
```

Gambar 4.25.2 Kondisi ketika utas tidak ditemukan

```
>> SAMBUNG_UTAS 2 1000;  
Index terlalu tinggi!
```

Gambar 4.25.3 Kondisi ketika index terlalu tinggi

4.26 Data Test 26

Memeriksa apakah pengguna dapat menghapus kicauan yang dimiliki pengguna dalam sebuah utas. Setelah memasukkan input yang diperlukan, kicauan dalam sebuah utas dengan ID utas dan index yang telah dimasukkan berhasil dihapus apabila ID utas ditemukan dan index valid.

```
Masukkan perintah  
>> HAPUS_UTAS 3 1;  
Kicauan sambungan berhasil dihapus!  
  
Ketik ';' untuk melanjutkan
```

Gambar 4.26.1 Kondisi ketika berhasil menghapus utas

4.27 Data Test 27

Memeriksa apakah output dapat menampilkan utas berdasarkan ID Utas. Setelah memasukkan ID utas, program akan mencetak seluruh kicauan dalam utas tersebut apabila ID utas ditemukan.

```
>> CETAK_UTAS;
3;
| ID = 3
| Tuan Hak
| 14/10/2023 11:10:1
| anjay mabar
|   | INDEX = 1
|   | Tuan Hak
|   | 24/11/2023 18:19:0
|   | PAKS A
|   | INDEX = 2
|   | Tuan Hak
|   | 24/11/2023 18:19:1
|   | AAA
|   | INDEX = 3
|   | Tuan Hak
|   | 24/11/2023 18:22:3
|   | INI SAMBUGNA
|
| Ketik ':' untuk melanjutkan
```

Gambar 4.27.1 Kondisi ketika berhasil mencetak utas

4.28 Data Test 28

Memeriksa apakah program dapat menyimpan kondisi dari BurBir saat ini ke dalam suatu folder berisi berkas-berkas konfigurasi. Setelah memasukkan nama folder penyimpanan, kondisi dari BurBir saat ini berhasil disimpan kedalam folder tersebut.

```

>> SIMPAN;
Masukkan nama folder penyimpanan
save1;
Folder save1 belum ada, akan dilakukan pembuatan folder terlebih dahulu.

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

Folder save1 berhasil dibuat.
Anda akan melakukan penyimpanan di folder save1.

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

Penyimpanan telah berhasil dilakukan!

```

Gambar 4.28.1 Kondisi ketika berhasil menyimpan file

4.29 Data Test 29

Memeriksa apakah program dapat memuat kondisi dari folder berisi berkas-berkas konfigurasi ke dalam aplikasi BurBir. Setelah memasukkan nama folder yang ingin di muat, program berhasil memuat kondisi dari folder berisi berkas-berkas konfigurasi ke dalam aplikasi BurBir apabila nama folder ditemukan.

```

Masukkan perintah
>> MUAT;|

```

Gambar 4.29.1

5 Test Script

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
1	Inisialisasi	Memeriksa apakah program dapat dimulai dan dapat membaca file config	Mengetik nama file config yang ingin dijalankan pada <i>command</i>	<i>Data Test 1</i>	Setelah memasukkan nama file config program berhasil berjalan	Sesuai yang ingin diharapkan Gambar 4.1.1

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
2	DAFTAR	Memeriksa apakah pengguna mendaftar dan terdaftar dengan nama unik beserta password antar pengguna	Mengetik <i>command</i> DAFTAR lalu memasukkan nama dan password	<i>Data Test 2</i>	Setelah memasukkan nama dan password, pengguna berhasil terdaftar dengan nama unik	Sesuai yang ingin diharapkan Gambar 4.2.1 Gambar 4.2.2
3	MASUK	Memeriksa apakah pengguna berhasil masuk dan menjalankan perintah lain dengan masukan nama dan password yang sesuai	Mengetik <i>command</i> MASUK lalu memasukkan nama dan password	<i>Data Test 3</i>	Setelah memasukkan nama dan password, pengguna berhasil masuk apabila memasukkan nama dan password dengan benar dan pengguna belum login	Sesuai yang ingin diharapkan Gambar 4.3.1 Gambar 4.3.2 Gambar 4.3.3
4	KELUAR	Memeriksa apakah pengguna berhasil keluar jika sebelumnya pernah login	Mengetik <i>command</i> KELUAR	<i>Data Test 4</i>	Setelah mengetik <i>command</i> KELUAR pengguna berhasil keluar apabila pengguna sudah login	Sesuai yang ingin diharapkan Gambar 4.4.1 Gambar 4.4.2
5	TUTUP_PROGRAM	Memeriksa apakah pengguna dapat keluar dari program BurBir	Mengetik <i>command</i> TUTUP_PROGRAM	<i>Data Test 5</i>	Setelah megetik <i>command</i> TUTUP_PROGRAM, pengguna berhasil keluar dari program BurBir	Sesuai yang ingin diharapkan Gambar 4.5.1
6	GANTI_PROFIL	Memeriksa apakah pengguna berhasil mengganti profil	Mengetik <i>command</i> GANTI_PROFIL lalu memasukkan profil yang baru	<i>Data Test 6</i>	Setelah memasukkan profil yang baru profil pengguna akan berganti dengan profil yang baru dimasukkan	Sesuai yang ingin diharapkan Gambar 4.6.1 Gambar 4.6.2

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
7	LIHAT_PR OFIL	Memeriksa apakah pengguna dapat melihat profil orang lain	Menetik <i>command</i> LIHAT_PROFIL diikuti nama pengguna yang ingin dilihat	<i>Data Test 7</i>	Setelah memasukkan nama pengguna yang ingin dilihat, output akan menampilkan profil pengguna tersebut apabila berjenis akun publik atau telah berteman dengan pengguna	Sesuai yang ingin diharapkan Gambar 4.7.1 Gambar 4.7.2
8	ATUR_JEN IS_AKUN	Memeriksa apakah pengguna dapat mengganti akunnya	Menetik <i>command</i> ATUR_JENIS_AKUN lalu ketik YA jika ingin mengubah dan TIDAK jika tidak	<i>Data Test 8</i>	Setelah menetik YA jenis akun pengguna berganti, namun jika menetik TIDAK jenis akun pengguna tidak berganti	Sesuai yang ingin diharapkan Gambar 4.8.1
9	UBAH_FOTO_PROFIL	Memeriksa apakah pengguna dapat mengubah foto profil	Menetik <i>command</i> UBAH_FOTO_PROFIL lalu masukkan foto profil yang baru	<i>Data Test 9</i>	Setelah memasukkan foto profil baru, foto profil pengguna akan berganti sesuai dengan yang baru dimasukkan	Sesuai yang ingin diharapkan Gambar 4.9.1
10	DAFTAR_TEMAN	Memeriksa output apakah dapat menampilkan daftar teman yang dimiliki pengguna	Menetik <i>command</i> DAFTAR_TEMAN	<i>Data Test 10</i>	Setelah menetik <i>command</i> DAFTAR_TEMAN, program akan menampilkan daftar teman yang dimiliki pengguna	Sesuai yang ingin diharapkan Gambar 4.10.1
11	HAPUS_TEMAN	Memeriksa apakah pengguna dapat	Menetik <i>command</i> HAPUS_TEMAN lalu masukkan nama teman	<i>Data Test 11</i>	Setelah memasukkan nama teman	Sedikit tidak sesuai dengan

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		menghapus teman yang dimiliki pengguna	yang ingin dihapus. Setelah itu ketik YA jika yakin dan TIDAK untuk sebaliknya		dan pengguna mengetik YA, program akan menghapus teman apabila sebelumnya pengguna saling berteman, Namun jika pengguna mengetik TIDAK program akan batal menghapus teman	yang diharapkan Gambar 4.11.1
12	TAMBAH_TEMAN	Memeriksa apakah pengguna dapat menambahkan teman	Mengetik <i>command</i> TAMBAH_TEMAN lalu masukkan nama pengguna yang ingin ditambahkan sebagai teman	<i>Data Test 12</i>	Setelah memasukkan nama pengguna, program akan mengirimkan permintaan pertemanan kepada pengguna tersebut apabila pengguna ditemukan	Sesuai yang ingin diharapkan Gambar 4.12.1
13	DAFTAR_PERMINTAAN_PERTEMANAN	Memeriksa apakah output dapat menampilkan permintaan teman yang dimiliki pengguna	Mengetik <i>command</i> DAFTAR_PERMINTAAN_PERTEMANAN	<i>Data Test 13</i>	Setelah mengetik <i>command</i> DAFTAR_PERMINTAAN_PERTEMANAN, program akan menampilkan daftar permintaan teman yang dimiliki pengguna berdasarkan kepopuleran	Sesuai yang ingin diharapkan Gambar 4.13.1

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
14	SETUJUI_PERTEMANAN	Memeriksa apakah pengguna dapat menyetujui permintaan pertemanan	Menetik <i>command</i> SETUJUI_PERTEMANAN. Setelah itu ketik YA jika menyetujui permintaan pertemanan dan ketik TIDAK jika tidak menyetujui	<i>Data Test 14</i>	Setelah menetik <i>command</i> SETUJUI_PERTEMANAN dan pengguna menetik YA maka pengguna berhasil menjalin pertemanan sebaliknya jika pengguna menetik TIDAK	Sesuai yang ingin diharapkan Gambar 4.14.1
15	KICAU	Memeriksa apakah pengguna dapat melakukan kicau	Menetik <i>command</i> KICAU lalu masukkan kicauan	<i>Data Test 15</i>	Setelah memasukkan kicauan, program akan mengirimkan detil kicauan	Sesuai yang ingin diharapkan Gambar 4.15.1
16	KICAUAN	Memeriksa apakah output dapat menampilkan kicauan dari pengguna ataupun dari teman pengguna	Menetik <i>command</i> KICAUAN	<i>Data Test 16</i>	Setelah menetik <i>command</i> KICAUAN program akan menampilkan detil kicauan dari pengguna ataupun dari teman pengguna	Sesuai yang ingin diharapkan Gambar 4.16.1
17	SUKA_KICAUAN	Memeriksa apakah pengguna dapat menyukai kicauan berdasarkan ID kicau	Menetik <i>command</i> SUKA_KICAUAN diikuti dengan ID kicau yang ingin disukai	<i>Data Test 17</i>	Setelah memasukkan ID kicau, pengguna berhasil menyukai kicauan apabila ID kicau ditemukan	Sesuai yang ingin diharapkan Gambar 4.17.1 Gambar 4.17.2
18	UBAH_KICAUAN	Memeriksa apakah pengguna dapat mengubah kicauan berdasarkan ID kicau	Menetik <i>command</i> UBAH_KICAUAN diikuti ID kicau yang ingin diubah. Jika ditemukan masukan kicauan yang baru	<i>Data Test 18</i>	Setelah memasukkan kicauan baru, kicauan berhasil diganti dengan kicauan yang baru	Sesuai yang diharapkan Gambar 4.18.1

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					dimasukkan apabila ID kicau ditemukan	
19	BALAS	Memeriksa apakah dapat Menambah balasan pada balasan dengan IDBalasan pada kicauan dengan IDKicau	Menetik <i>command</i> BALAS diikuti dengan ID Kicau dan ID Balasan	<i>Data Test 19</i>	Setelah input id kicau dan id balasan, jika id kicau dan id balasan valid maka balas akan ditambahkan ke id yang dituju	Sesuai yang ingin diharapkan Gambar 4.19.1 Gambar 4.19.2
20	BALASAN	Memeriksa apakah output dapat menampilkan daftar balasan pada kicau	Menetik <i>command</i> BALASAN diikuti ID Kicau yang ingin dilihat	<i>Data Test 20</i>	setelah memasukan id kicau, jika id valid akan ditampilkan seluruh balasan dalam sesuai kicauan tersebut menampilkan dengan sistem print DFS	Sesuai yang ingin diharapkan Gambar 4.20.1 Gambar 4.20.2
21	HAPUS_BALASAN	Memeriksa apakah pengguna dapat menghapus balasan	Menetik <i>command</i> HAPUS_BALASAN diikuti dengan ID Kicau dan ID Balasan	<i>Data Test 21</i>	setelah memasukan id kicau dan id balasan, jika valid maka seluruh balasan dengan id yang dipilih akan terhapus	Sesuai yang ingin diharapkan Gambar 4.21.1 Gambar 4.21.2
22	BUAT_DRAFT	Memeriksa apakah pengguna dapat membuat draf	Menetik <i>command</i> BUAT_DRAFT lalu masukkan draf. Pengguna perlu memasukkan HAPUS jika ingin menghapus draf, SIMPAN jika ingin menyimpan draf, dan TERBIT jika ingin menerbitkan draf.	<i>Data Test 22</i>	user memasukan isi draft, lalu memilih perintah, jika memilih simpan draft akan disimpan dalam stack, jika memilih terbit maka draft akan	Sesuai yang diharapkan Gambar 4.22.1 Gambar 4.22.2

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					langsung masuk ke kicau, jika memilih hapus tidak terjadi apa dan draft tidak tersimpan	
23	LIHAT_DR AF	Memeriksa apakah output dapat menampilkan draf yang dimiliki pengguna	Menetik <i>command</i> LIHAT_DRAF	<i>Data Test 23</i>	saat mulai fungsi, jika user memiliki draft maka top dari stack akan ditampilkan, draft dapat diubah, simpan dan dihapus.	Sesuai yang diharapkan Gambar 4.23.1 Gambar 4.23.2
24	UTAS	Memeriksa apakah pengguna dapat membuat UTAS	Menetik <i>command</i> UTAS diikuti dengan ID Kicau lalu masukkan kicauan. Ketika selesai, pengguna perlu memasukkan YA; jika ingin melanjutkan utas dan TIDAK; jika ingin berhenti.	<i>Data Test 24</i>	Setelah memasukkan input yang diperlukan, utas baru akan berhasil dibuat dengan kicauan pertamanya adalah kicauan dengan ID yang dimasukkan apabila ID kicau ditemukan	Sesuai yang diharapkan Gambar 4.24.1 Gambar 4.24.2
25	SAMBUNG_UTAS	Memeriksa apakah pengguna dapat menambahkan kicauan baru dalam sebuah utas dengan ID Utas dan pada posisi index	Menetik <i>command</i> SAMBUNG_UTAS diikuti dengan ID Utas dan index	<i>Data Test 25</i>	Setelah memasukkan ID utas dan index, sebuah kicauan baru berhasil ditambahkan dalam sebuah utas apabila ID utas ditemukan dan index valid	Sesuai yang diharapkan Gambar 4.25.1 Gambar 4.25.2 Gambar 4.25.3
26	HAPUS_UTAS	Memeriksa apakah pengguna	Menetik <i>command</i> HAPUS_UTAS diikuti dengan ID Utas dan index	<i>Data Test 26</i>	Setelah memasukkan input yang	Sesuai yang diharapkan

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
		dapat menghapus kicauan yang dimiliki pengguna dalam sebuah utas	sesuai dengan kicauan yang ingin dihapus		diperlukan, kicauan dalam sebuah utas dengan ID utas dan index yang telah dimasukkan berhasil dihapus apabila ID utas ditemukan dan index valid	Gambar 4.26.1
27	CETAK_UTAS	Memeriksa apakah output dapat menampilkan utas berdasarkan ID Utas	Menetik <i>command</i> CETAK_UTAS diikuti dengan ID Utas	<i>Data Test 27</i>	Setelah memasukkan ID utas, program akan mencetak seluruh kicauan dalam utas tersebut apabila ID utas ditemukan	Sesuai yang diharapkan Gambar 4.27.1
28	SIMPAN	Memeriksa apakah program dapat menyimpan kondisi dari BurBir saat ini ke dalam suatu folder berisi berkas-berkas konfigurasi	Menetik <i>command</i> SIMPAN lalu masukkan nama folder penyimpanan	<i>Data Test 28</i>	Setelah memasukkan nama folder penyimpanan, kondisi dari BurBir saat ini berhasil disimpan kedalam folder tersebut	Sesuai yang diharapkan Gambar 4.28.1
29	MUAT	Memeriksa apakah program dapat memuat kondisi dari folder berisi berkas-berkas konfigurasi ke dalam aplikasi BurBir	Menetik <i>command</i> MUAT lalu masukkan nama folder yang ingin dibuat	<i>Data Test 29</i>	Setelah memasukkan nama folder yang ingin di muat, program berhasil memuat kondisi dari folder berisi berkas-berkas konfigurasi ke dalam aplikasi BurBir apabila	Sedikit tidak Sesuai yang ingin diharapkan Gambar 4.29.1

No.	Fitur yang Dites	Tujuan Testing	Langkah-Langkah Testing	Input Data Test	Hasil yang Diharapkan	Hasil yang Keluar
					nama folder ditemukan	

6 Pembagian Kerja dalam Kelompok

NIM	Nama	Pembagian Kerja
13522134	Shabrina Maharani	ADT datetime, ADT kicauan, driver kicauan, daftar, masuk, keluar.
13522147	Ikhwan Al Hakim	Implementasi word machine, ADT Utas, driver utas, empty utas, isi utas, cetak utas, sambung utas, hapus utas, merging dan fix semua fungsi ke main.
13522152	Muhammad Roihan	ADT matrix, driver profil, ganti profil, lihat profil, atur jenis akun, ubah foto profil
13522159	Rafif Ardhinto Ichwantoro	ADT Balasan, ADT draftkicau, driver Balasan, driver draftkicau.
13522161	Mohammad Akmal Ramadan	ADT graph, Makefile, driver teman, fungsi daftar teman, hapus teman, daftar permintaan pertemanan, dan setuju pertemanan

7 Lampiran

7.1 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!

7.2 Notulen Rapat

Hari, Tanggal	NIM	Notulen Rapat
Rabu, 25 Oktober 2023	13522134 13522147 13522152 13522159 13522161	Diskusi pembagian kerja pertama.
Senin, 30 Oktober 2023	13522134 13522147 13522152 13522159	Progress pengerjaan tugas besar

	13522161	
Rabu, 1 November 2023	13522134 13522147 13522152 13522159 13522161	Asistensi I
Jumat, 10 November 2023	13522134 13522147 13522152 13522159 13522161	Progress pengerjaan tugas besar
Selasa, 14 November 2023	13522134 13522147 13522152 13522159 13522161	Asistensi II
Kamis, 16 November 2023	13522134 13522147 13522152 13522159 13522161	Progress pengerjaan tugas besar
Senin, 20 November 2023	13522134 13522147 13522152 13522159 13522161	Progress pengerjaan tugas besar
Jumat, 24 November 2023	13522134 13522147 13522152 13522159 13522161	Progress pengerjaan tugas besar

7.3 Log Activity Anggota Kelompok

Shabrina Maharani / 13522134

30 Oktober 2023 : Membuat ADT pengguna dengan fungsi daftar, masuk, keluar
1 November 2023 : Memperbaiki ADT datetime
3 November 2023 : Membuat ADT kicauan

8 November 2023 : Membuat ADT listdinkicauan
12 November 2023 : Menggabungkan ADT listdinkicauan ke ADT kicauan
18 November 2023 : Membuat driver kicauan

Ikhwan Al Hakim / 13522147

27 Oktober 2023 : Membuat implementasi word machine
29 Oktober 2023 : Membuat ADT Utas dan semua fungsi utas
2 November 2023 : Membuat driver utas
12 November 2023 : Mulai penggabungan dan fixing ke main (sampai hari H deadline)

Muhammad Roihan / 13522152

2 November 2023 : Membuat ADT matrix, fungsi ganti profil, dan ubah foto profil
4 November 2023 : Membuat fungsi atur jenis akun
14 November 2023 : Membuat fungsi lihat profil
18 November 2023 : Membuat driver profil

Rafif Ardhinto Ichwantoro / 13522159

3 November : Membuat Adt tree dan stack
8 November : Membuat Balasan
15 November : Membuat ADT driveKicauan
21 November : membuat driver balasan dan drivekicauan

Mohammad Akmal Ramadan / 13522161

28 Oktober 2023 : Membuat ADT graph
30 Oktober 2023 : Membuat daftar teman dan hapus teman
1 November 2023 : Membuat daftar permintaan pertemanan
2 November 2023 : Membuat terima pertemanan
6 November 2023 : Membuat Makefile
12 November 2023 : Membuat driver teman

7.4 Form Asistensi

**Form Asistensi Tugas Besar
IF2110/Algoritma dan Struktur Data
Sem. 1 2023/2024**

No. Kelompok/Kelas : H/K3
Nama Kelompok : HAHAAH
Anggota Kelompok (Nama/NIM) :
1. Shabrina Maharani/13522134
2. Ikhwan Al Hakim/13522147
3. Muhammad Roihan/13522152
4. Rafif Ardinto Ichwantoro/13522159
5. Mohammad Akmal Ramadan/13522161







Asisten Pembimbing : Hafidz Nur Rahman Ghozali

Form Asistensi Tugas Besar
IF2110/Algoritma dan Struktur Data
Sem. 1 2023/2024







No. Kelompok/Kelas : H/K3
Nama Kelompok : HAHAAH
Anggota Kelompok (Nama/NIM) :
1. Shabrina Maharani/13522134
2. Ikhwan Al Hakim/13522147
3. Muhammad Roihan/13522152
4. Rafif Ardhinto Ichwantoro/13522159
5. Mohammad Akmal Ramadan/13522161

Asisten Pembimbing : Hafidz Nur Rahman Ghozali

Asistensi I

Tanggal : 01/11/2023	Catatan Asistensi:
Tempat : Zoom Meeting	Pembagian penggunaan ADT untuk setiap fungsinya
<p>Kehadiran Anggota Kelompok:</p> <p>No</p> <p>NIM</p> <p>Tanda tangan</p> <p>1</p> <p>13522134</p>  <p>2</p> <p>13522147</p>  <p>3</p> <p>13522152</p>  <p>4</p> <p>13522159</p>  <p>5</p> <p>13522161</p> 	<p>menjadi sebagai berikut:</p> <ul style="list-style-type: none"> - Pengguna menggunakan ADT buatan sendiri - Profil menggunakan ADT matriks untuk foto profilnya - Teman menggunakan ADT graf - Permintaan pertemanan menggunakan ADT stack (agak lupa) - Kicauan menggunakan ADT tree (agak lupa) - Balasan juga menggunakan ADT tree karena berhubungan dengan kicauan - Draf kicauan menggunakan ADT stack - Utas menggunakan ADT linked list <p>Selain itu pada asistensi tadi kami juga menanyakan berbagai hal yang masih kurang jelas pada spesifikasi seperti konfigurasi yang kurang lengkap pada spek.</p>
	<p>Tanda Tangan Asisten:</p> 

Asistensi II

Tanggal : 14/11/2023	Catatan Asistensi:
Tempat : Zoom Meeting	Progress sudah bagus, tetapi masih ada beberapa hal yang kurang yaitu:
<p>Kehadiran Anggota Kelompok:</p> <p>No NIM Tanda tangan</p> <p>1 13522134</p>  <p>2 13522147</p>  <p>3 13522152</p>  <p>4 13522159</p>  <p>5 13522161</p> 	<ul style="list-style-type: none"> - Pembuatan driver masing-masing ADT untuk unit-testing - Pembuatan MakeFile biar nggak perlu capek-capek copy paste commandline setiap mau compile - Perbaiki pembacaan folder config biar bisa membaca nama folder dan nama config apapun - Struktur program mungkin dirapiin lagi
	<p>Tanda Tangan Asisten:</p> 

7.5 Milestone

Milestone 1 Kelompok H

Anggota Kelompok: - Shabrina Maharani 13522134
- Ikhwan Al Hakim 13522147
- Muhammad Roihan 13522152
- Rafif Ardhinto Ichwantoro 13522159
- Mohammad Akmal Ramadan 13522161

No	Fitur	Progress (%)	Keterangan
1	Inisialisasi	100	Pembuatan main awal sudah selesai dan bug pembacaan config kemarin sudah di fix :)
2	Perintah	50	-
3	Pengguna	100	-
4	Profil	80	-
5	Teman	60	-
6	Permintaan Pertemanan	50	-
7	Kicau	50	-

8	Balasan	50	-
9	Draf Kicauan	50	-
10	Utas	60	-
11	Simpan dan Muat	0	Belum dibuat sama sekali karena semua program harus udah jadi dulu biar ada yang disimpan
12	Laporan	0	Belum dibuat sama sekali karena laporan bisa kelar seminggu (bismillah)